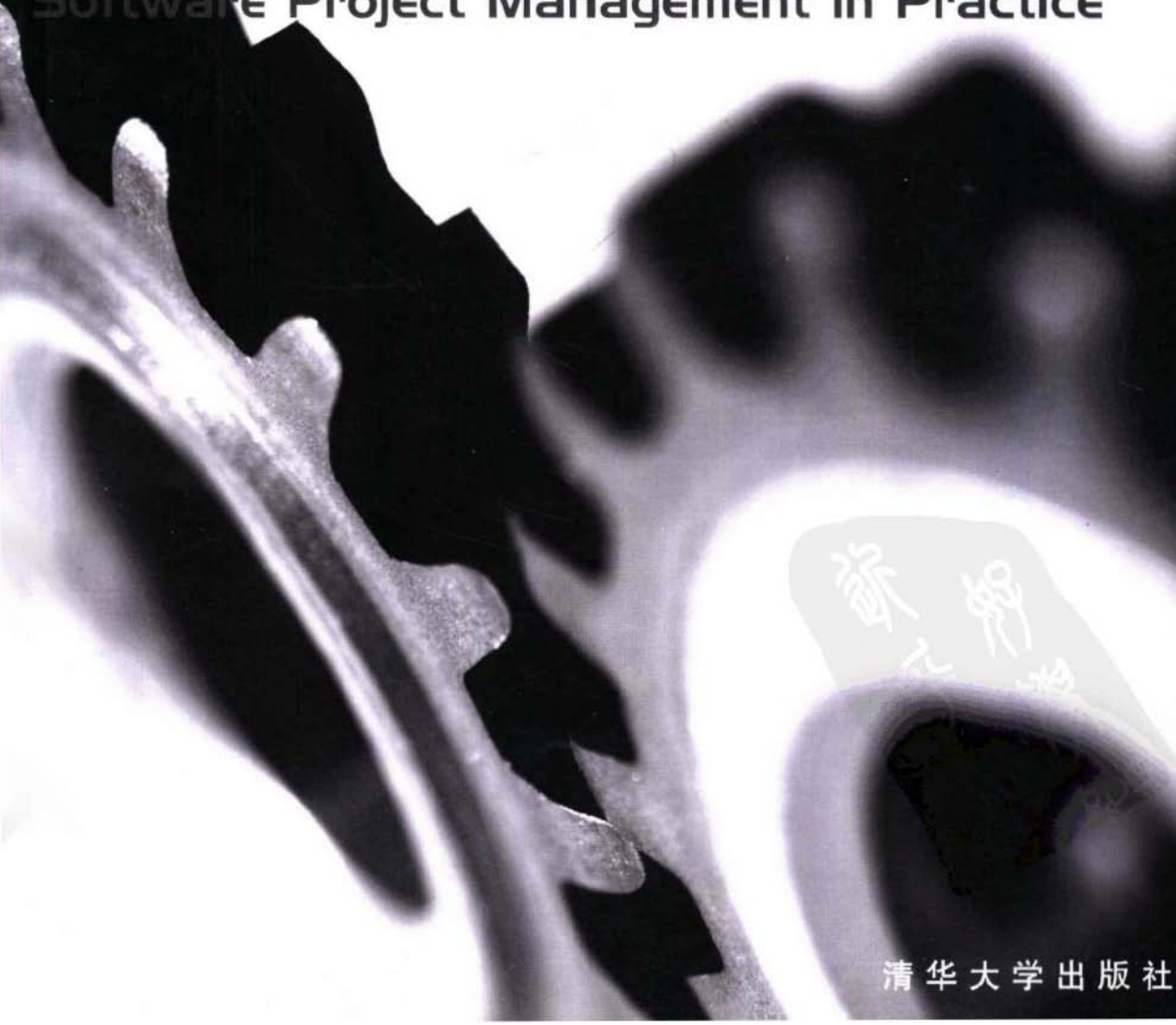
PEARSON
Addison Wesley

软件工程实践丛书 软件学院教材

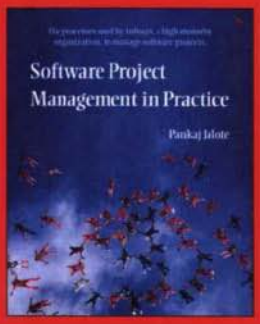
软件项目管理实践

[印] Pankaj Jalote 著 施平安 译 左美云 审

Software Project Management in Practice



清华大学出版社



软件项目管理实践

内容简介：

我们经常听说软件项目推迟交货、超出预算或者不能满足客户需要。然而，一些组织能够成功地管理一个又一个满足期望结果的项目。在本书中，Pankaj Jalote讨论了这样一个公司——Infosys技术公司，这是一个著名的高成熟度组织；书中详细讨论了它管理项目的过程。Jalote完全揭示了Infosys是如何运作的，提供了一个优秀的案例研究以指导所有的项目经理。本书描述的特定的Infosys实践反映了合理的管理原则和实践。它们也植根于常识，并且可以容易地与任何组织的软件开发活动结合。

本书所讨论的实践与广泛接受的能力成熟度模型（CMM）也是一致的。最后，读者将得到一个系统地改进任何软件项目的规划和执行的实用框架。

本书可作为软件学院及大学计算机等专业相关课程的教材，也可以作为软件公司各级管理和开发人员参考。

作者简介：

Pankaj Jalote是印度理工大学计算机科学与工程系的教授和系主任。曾经是马里兰大学计算机科学系的副教授，担任过印度班加罗尔的Infosys技术有限公司质量部门的副总裁。在Infosys公司任职期间，他是推动Infosys向CMM高成熟度等级发展的主要设计师之一。他也是《CMM实践：Infosys执行软件项目的过程》一书的作者，他在该书中讨论CMM在一个组织中的实施。

ISBN 7-302-06392-3

9 787302 063926 >

定价：36.00元

PEARSON
Addison
Wesley

责任编辑：尤晓东
封面设计：立日新设计公司
读者信箱：Book@21bj.com
信息网站：http://www.ePress.com.cn

软件工程实践丛书

软件项目管理实践

[印] Pankaj Jalote 著
施平安 译
左美云 审

清华大学出版社
·北京·

内 容 简 介

本书以 Infosys 公司的真实软件项目为例，阐明了成功的软件项目管理涉及的关键特征。项目规划部分讨论了规划基础结构、过程规划、工作量和进度估计、质量规划、风险管理、度量规划以及配置管理等主题；项目执行和收尾部分讨论了项目评审、项目监督和控制以及项目收尾等主题。全书提供了一个有助于系统地改进软件项目的规划和执行的实用框架。本书讨论的实践与当前广泛接受的能力成熟度模型（CMM）是一致的。

本书可作为软件学院及大学相关专业本科生和研究生的教材，也适合项目经理和打算成为项目经理的专业人士阅读。

EISBN: 0-201-73721-3
Software Project Management in Practice
Pankaj Jalote
Copyright © 2002 by Pearson Education, Inc.
Orlginal English language edition published by Pearson Education, Inc.
All right reserved.

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR),**仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。**

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

北京市版权局著作权合同登记号：图字 01-2002-4424 号

图书在版编目（CIP）数据

软件项目管理实践 /（印）杰罗特著；施平安译． —北京：清华大学出版社，2003
（软件工程实践丛书）
书名原文：Software Project Management in Practice
ISBN 7-302-06392-3
Ⅰ.软… Ⅱ.①杰… ②施… Ⅲ.软件开发—项目管理 Ⅳ.TP311.52
中国版本图书馆 CIP 数据核字（2003）第 018781 号

出 版 者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
http://www.tup.com.cn
http://www.tup.tsinghua.edu.cn
责任编辑：尤晓东
印 刷 者：世界知识印刷厂
发 行 者：新华书店总店北京发行所
开 本：787×960 1/16 印张：13.75 字数：270 千字
版 次：2003 年 4 月第 1 版 2003 年 4 月第 1 次印刷
书 号：ISBN 7-302-06392-3/TP・4819
印 数：0001～5000
定 价：36.00 元

译 者 序

全球每年要完成上百万个软件项目，但有 1/3 左右的项目在成本和时间上都超出额定限度的 125%以上。为什么这么多的软件项目会失败呢？关键问题在于软件项目管理不正确。例如，目标不明确、计划不完善、采用新技术、缺乏项目管理方法，以及人员不够等等。因此，必须改进项目的执行过程，采用有效的项目管理技术，这样软件项目成功的机会就会大大地提高。本书以 Infosys 公司的真实软件项目为例，阐明了成功的软件项目管理涉及的关键特征。全书提供了一个有助于系统地改进软件项目的规划和执行的实用框架。本书讨论的实践与当前广泛采用的能力成熟度模型（CMM）是一致的。

本书作者潘克杰·杰罗特（Pankaj Jalote）是印度理工大学计算机科学与工程系的教授，曾经是马里兰大学计算机科学系的副教授，担任过印度班加罗尔的 Infosys 技术有限公司质量部门的副总裁。在 Infosys 公司任职期间，他是推动 Infosys 向 CMM 高成熟度等级发展的主要设计师之一。

本书在内容组织上尽量使各章保持独立，互不依赖。首先介绍了 Infosys 公司以及与项目管理有关的 CMM 内容，然后按项目规划、项目执行和收尾两部分展开全书的介绍。项目规划（第 2 章到第 9 章）部分讨论了规划基础结构、过程规划、工作量和进度估计、质量规划、风险管理、度量规划以及配置管理等主题；项目执行和收尾（第 10 章到第 12 章）部分讨论了项目评审、项目监督和控制以及项目收尾等主题。

本书的主要读者对象为项目经理和打算成为项目经理的专业人士。他们可以运用本书中描述的方法，系统地改进项目的规划和执行。本书对于想要达到一个高成熟度等级的组织也是非常有用的。只要项目经理采用本书中描述的方法，他们就可以为组织通往更高的能力成熟度等级奠定基础。此外，本书也可以作为软件项目管理课程的教科书。

全书由我一人翻译，施惠琼负责全书的录入和审校工作，在此对她表示感谢。我还要感谢我的家人在我翻译本书中表现出来的无限耐心，感谢他们为我所提供的帮助和支持。尤其是我的女儿，她那天真可爱的笑声给予了我无比的力量。没有他们的谅解和支持，很难想象能够在这么短的时间内完成本书的翻译工作。

在翻译过程中，我对本书中出现的所有术语和难词难句都进行了仔细的推敲和研究，然而有些方面在译者本人的研究领域中也不曾遇到过，疏漏和争议之处在所难免，望广大读者提出宝贵的意见。

施平安

2002/11/5 于海军广州舰艇学院

前　言

本书的写作起因可以追溯到 1996 年。我在休假年加入了 Infosys 公司，被任命为质量主管，负责改进项目的执行过程。看到项目经理所面临的问题，我感到软件项目管理必定是所有工作中最为棘手的。当时我认为有限的预算与按时交付高质量的软件必定相互矛盾。

现在，我意识到这种情况也并非完全没有解决的可能性。通过使用合理的项目管理过程，一些组织中的项目经理正在成功地执行项目。

Infosys 公司就是这样的一个全球性组织，仅去年一年，它就成功地执行了 500 多个项目。该公司大约有 10 000 名员工，分布在全球十几个国家的 25 个城市中，经评估它已达到了 CMM 最高成熟度等级（第 5 级）。在过去的 5 年中，其年收入和利润以每年 75%左右的速度增长。在公司的巨额增长中，85%的收入来自现有客户的后续订单，因此可以判断出客户对公司的满意度是很高的。

根据在 Infosys 公司管理过程的经验和知识，我精选出了那些已经成功地执行了的主要项目管理实践，并在本书中进行了描述。这些实践的优点在于：它们虽然极为有效，但并不复杂；更确切地讲，它们植根于常识，并由简单的度量和分析提供支持。

在对 Infosys 公司的项目管理过程进行描述的过程中，本书提供了两种价值观的独特组合。首先，它提供了在一个非常成功的组织中项目管理所采用的完整的过程集。这些过程使用真实的例子和一个贯穿本书始末的研究案例加以阐述。其次，因为这些过程满足能力成熟度模型（Capability Maturity Model，简称 CMM）的需求，所以本书阐明了高成熟度组织的项目管理方法，并介绍了 CMM 为项目管理带来的好处，但有关人员不必完全理解 CMM。

本书的主要读者对象为项目经理和打算成为项目经理的专业人士。通过采用本书描述的方法，他们可以系统地改进项目的规划和执行。本书对于想要达到一个高成熟度等级的组织也是非常有用的。如果项目经理开始使用本书描述的方法，他们将为组织的高成熟度奠定基础。

因为在描述 Infosys 方法的具体内容之前，大部分章节都以一个综述开始，所

以也可以把本书作为软件项目管理课程的教科书。在一般的软件工程课程中，本书可以作为辅助教材，提供了如何在一个商业环境中管理软件项目的思路。

第 1 章简要介绍了 Infosys 公司以及 CMM 与项目管理的关系。其余内容分为两部分。第 1 部分从第 2 章到第 9 章，重点是项目规划，并讨论了诸如规划基础结构、过程规划、工作量和进度估计、质量规划、风险管理、度量规划以及配置管理等主题。第 2 部分从第 10 章到第 12 章，重点是项目执行和收尾，并讨论了评审、项目监督和控制以及项目收尾等主题。

各章尽量保持独立，互不依赖，并按如下顺序进行组织：首先提供相关的概念和背景材料，随后讨论 Infosys 公司所用的方法；接着讲述这些方法在实际项目和案例研究中的使用实例。每章都以一个小结收尾，该小结列出了本章讨论的要点，并归纳了本章所讨论的方法满足了 CMM 的哪些特征。

虽然本书基于我在早些时候撰写的《CMM 实践》（*CMM in Practice*，Addison-Wesley 出版社，2000 年出版）一书，但是他们的侧重点是不同的，并且本书还包含大量其他内容。鉴于《CMM 实践》考虑整个软件过程并侧重于CMM 在组织中的实现，本书专门讲述项目管理。

本书的出版得到了很多人的帮助。因为本书源自《CMM 实践》，所以，感谢为那本书的撰写提供过帮助的所有人。此外，再次感谢 Infosys 公司及其主管，他们的合作和帮助使本书得以顺利完成。衷心感谢 Infosys 公司质量部门的成员，在需要的时候，他们提供了所需的信息；感谢众多与我分享他们的经验的人，他们努力形成了贯穿本书始末的小型案例（尽管改变了案例名字）。特别感谢 Naresh Agarwal，感谢他在主要的案例研究上提供的帮助；感谢 Sanjay Joshi，他使得每章内容的重点更加突出。最后，感谢我的妻子 Shikha 和女儿 Sumedha 和 Sumanda，感谢他们再次对我表现出的宽容。

有关本书的任何评论或者可能存在的任何不正确之处（这完全是我的责任），请发送到我的电子邮箱 jalote@iitk.ac.in。有关 Infosys 公司的信息，请访问网站 http://www.infy.com 或者发送电子邮件到 publicrelations@infy.com 了解。

目 录

第 1 章 软件项目管理 1
1.1 过程与项目管理 2
1.2 项目管理与 CMM 4
1.3 Infosys 公司的项目管理 8
1.4 ACIC 案例研究综述 13
1.5 小结 14
1.6 参考文献 15

第 1 部分 项目规划

第 2 章 项目规划基础结构 17
2.1 过程数据库 18
2.2 过程能力基准 21
2.3 过程资源和知识库系统 24
2.4 小结 26
2.5 参考文献 27

第 3 章 过程规划 28
3.1 Infosys 开发过程 28
3.2 需求变更管理 34
3.3 ACIC 项目的过程规划 38
3.4 小结 39
3.5 参考文献 39

第 4 章 工作量估计和进度计划 41
4.1 估计和进度安排概念 41
4.2 工作量估计 44
4.3 进度计划 52
4.4 小结 60
4.5 参考文献 60

第 5 章 质量规划 62
5.1 质量概念 62
5.2 定量化质量管理规划 66
5.3 故障预防规划 68
5.4 ACIC 项目的质量计划 71
5.5 小结 73
5.6 参考文献 74
第 6 章 风险管理 76
6.1 风险和风险管理概念 77
6.2 风险评估 80
6.3 风险控制 83
6.4 实例 85
6.5 小结 87
6.6 参考文献 88
第 7 章 计划的度量和跟踪 89
7.1 度量概念 89
7.2 度量 93
7.3 项目跟踪 98
7.4 ACIC 项目的度量和跟踪计划 100
7.5 小结 101
7.6 参考文献 101
第 8 章 项目管理计划 103
8.1 团队管理 103
8.2 客户沟通和问题解决方案 105
8.3 项目管理计划的结构 106
8.4 ACIC 项目计划 107
8.5 小结 121
第 9 章 配置管理 122
9.1 配置管理概念 123
9.2 配置管理过程 126
9.3 ACIC 项目的配置管理计划 131
9.4 小结 137
9.5 参考文献 137

第 2 部分 项目执行和收尾

第 10 章 评审 139
10.1 评审过程 139
10.2 数据收集 144
10.3 监督和控制 147
10.4 评审和 NAH（这里不适用）综合症介绍 152
10.5 小结 156
10.6 参考文献 157

第 11 章 项目监督和控制 158
11.1 项目跟踪 159
11.2 里程碑分析 165
11.3 用 SPC（统计过程控制）进行任务级分析 171
11.4 故障分析和预防 172
11.5 过程监督和审计 181
11.6 小结 185
11.7 参考文献 185

第 12 章 项目收尾 187
12.1 项目收尾分析 187
12.2 ACIC 项目的收尾分析报告 191
12.3 小结 199
12.4 参考文献 199

第 1 章　软件项目管理

全世界每年大约有 50 万个项目经理执行着 100万个左右的软件项目，产生了价值 6 000 亿美元的软件产品。在这些项目中，有很多不能满足客户所期望的质量，或者不能在预算内按时交付软件。有分析认为：1/3 左右的项目在成本和时间上超出额定限度的 125%以上。[1]

为什么这么多的软件项目会失败呢？失败的原因虽然很多，但最重要的一条是软件项目管理不正确。例如，项目失控的主要原因有目标不明确、计划不完善、新技术、缺乏项目管理方法论以及人员不够。[2] 显然，其中至少有 3 个原因与项目管理有关。另外 2 个原因——人员不够和新技术，可以说是风险活动，但风险活动的管理也是项目管理的一部分。

显然，通过采用有效的项目管理技术，项目经理可以提高项目成功的机会。但这些有效的技术是什么呢？

我们可以考虑一个类似的例子。假定你想炼就一个肌肉发达而强健的体魄。为了达到此目的，你开始查找杂志上描述的练习规程。有篇文章描述了如何锻炼手臂力量，给出了一套 10 个练习的方案——无论如何这些练习也不能算太多。然而，另一篇关于锻炼大腿力量的文章也给出了 10 个练习，而一般人仍然会感到做 10 个练习不算太多。如果你想根据这些不同的练习计划来锻炼全身，你将会发现需要做 50 到 100个练习——显然，这对于大多数人是不能接受的，更不用说繁忙的项目经理了。为了实现目标，你需要有一个实用而高效的综合训练计划。

同样，你也可能会发现大量关于执行各方面项目管理的建议，包括工作量估计、风险管理、项目监督、配置管理等等。虽然每种建议的技术都解决了它计划要解决的问题，但是如何将这些技术组合成一个切实可行的项目管理过程却是不明确的。为了实现有效的项目管理，迫切需要的是能够产生良好结果的、实用的、可管理的“练习例程（exercise routine）”。换句话说，我们所需要的是一个能够对从开始到收尾的整个项目进行管理的平衡过程。遗憾的是，只有少数公开的方法说明了如何以这种方法集成项目管理技术。

本书填补了这一空白，它描述了在一个世界一流的组织中使用的一套高效而有用的管理软件项目的过程。这个公司就是 Infosys，它是一家软件开发公司，该

公司在软件项目运作方面有着令人羡慕的记录：仅在 2000 年，Infosys 公司的项目经理就用本书描述的过程成功地为客户开发了约 500 个项目。本书讨论了 Infosys 公司项目管理的方方面面——项目规划、执行和收尾。你将知道 Infosys 公司的项目经理如何估计工作量、规划风险管理、收集度量数据、设置质量目标以及使用监督项目的度量指标等等。对繁忙的项目经理最有吸引力的是，这些过程既不复杂也不麻烦，而且它们使用简单的度量指标。

Infosys 公司已经通过了能力成熟度模型（Capability Maturity Model，简称 CMM）第 5 级 CMM-5 的评估，CMM-5 是 CMM 中的最高级。本书不仅从 Infosys 公司的一系列过程中提炼出了项目管理过程，而且还说明了在一个高成熟度的组织中如何对项目进行管理。通过说明，我希望 CMM 的好处能够引起那些没有时间对它进行研究的项目经理的注意，因为他们把它看成“过程族（process folks）”，或者因为他们发现很难将 CMM 与项目管理实践联系起来。

本章介绍了两个主题，它们是本书的背景知识：CMM 模型和 Infosys 公司。因为本书的重点是项目管理而不是 CMM，所以仅讨论 CMM 的项目管理方面的内容。本章还概述了项目管理过程，提供了主要案例研究；有关这些讨论的详细内容将在后文进行介绍。因此，让我们先来简单讨论一下过程在项目管理中的作用。

1.1 过程与项目管理

软件项目主要涉及两方面的任务：软件工程和项目管理。软件工程方面涉及系统的建立，并重点关注如何设计、测试、编码等问题。项目管理方面涉及如何正确地规划和控制软件工程行为，以满足项目在成本、进度和质量方面的目标。

如果项目的规模较小（比如说由 1 到 2 个人组成的团队工作几个星期的项目），则可以用不太正规的方法实现它。项目计划可能是一个电子邮件，它规定了项目的交付日期，另外还有可能规定了一些中间里程碑。需求可能通过一个便条甚至口头方式进行通知，而中间工作产品（诸如设计文档）可能是在个人的便条本上草草写就的东西。

然而，对于那些需要很多人工作很多个月才能完成的大型项目，这些不正规的技术却是不适用的——大多数商业软件项目都存在这种情况。在这样的项目中，必须遵循那些经试验证明效果良好的方法谨慎地执行每个工程任务，并且必须用

文档正确地记录工作产品，以便其他人员能够查阅它们。项目任务必须经过仔细规划，并将它们分配给执行项目的人员，然后在项目执行的同时对它们进行跟踪。换句话说，要成功地执行较大规模的项目，必须在软件工程和项目管理这两方面增强正规程度和严格程度。

正规方法要求用良好定义的过程来执行各种任务，以便结果更加依赖于过程的能力。如果通过使用合适的度量手段在过程中实现了量化方法，则正规程度得到进一步的加强。

那么，何谓过程呢？从技术上讲，对于某个任务的一个过程，它由执行该任务时应当遵循的一个步骤序列组成。然而，对一个组织而言，它建议其工程师和项目经理使用的过程就远非一个步骤序列了；它们涵盖了工程师们和项目经理所掌握的成功地执行项目的实践。通过过程就可以将好的经验同所有人共享，包括公司中新来的员工。这些过程有助于管理人员和工程师借鉴过去的成功经验，从而避开导致失败的陷阱。

对于一个项目而言，软件工程过程通常规定如何执行工程任务，诸如需求规范、设计、测试等等；而项目管理过程则规定如何设置里程碑、组织全体人员、管理风险、监督进展等任务。本书重点介绍项目管理过程。

当你考虑项目管理过程时，你必须弄清楚项目经理是否会采用它们。我经常听到程序设计人员抱怨项目经理不遵循过程并抵制变化。我在 Infosys 及其他公司担当项目经理的经验告诉我，只有在项目管理过程合理并有助于项目经理更好地执行他们的项目时，他们才会真正地想到采用过程。然而，对于那些看起来一味地墨守成规并且于他们的工作毫无价值的过程，项目经理确实会感到很气愤。因此，较好的做法是制定轻型过程（light weight）——那些有助于项目经理更好地规划和控制他们的项目并使他们能够灵活地处置各种情况的过程。

为了回答“为什么项目经理必须遵循过程？”Infosys 公司的创立者、董事及现任客户中心主管 S.D. Shibulal，他用如下几个要点进行了精辟的总结：

- 过程代表着集体的智慧，使用它们可以增加成功的机会。
- 过程可能会包含一些多余的步骤，但是你事先不可能完全知道哪些步骤是不必要的，因此走捷径可能会增加风险。
- 如果没有采用过程，你就不能很好地预测项目的结果。
- 如果没有定义过程，你和组织就不能进行有效地学习。而学习和提高是当今知识世界必不可少的事情。

- 过程减少了你须考虑的问题。检查表（checklist）必然会涵盖 80%要做的事情，大大地减少了你的任务，因而你只须完成剩下的20%。

1.2 项目管理与 CMM

一旦承认有效的过程有助于成功地执行一个项目，必然会引出这样一个问题：这些过程有哪些必不可少的特征呢？软件 CMM 提出了解决这个问题的框架。

软件 CMM 是由卡内基·梅隆大学的软件工程研究所（Software Engineering Institute，简称 SEI）开发的，它体现了软件组织和其他组织在软件开发管理方面的最佳实践。因此，CMM 体现了集体的过程经验及很多公司的期望。它规定了过程所必需的特征，但是没有规定专门的过程。因此，不同的过程都可以实现CMM的要求。CMM 可以用来评价一个组织的软件过程，标识出存在的缺陷。

CMM 是目前使用最为普遍的软件过程改进框架之一（现在其他常用框架有 ISO 9001[3,4,5]）。瓦茨·汉弗莱的《软件过程管理》奠定了 CMM 的基础[6]，而框架本身在 SEI 的《能力成熟度模型：软件过程改进指南》中进行了详细的描述。[7]CMM 的最新版本，CMM-I，已经发布了。然而，由于本书的重点不在于 CMM 模型，再者 CMM-I 可用的经验仍然很有限，因此我们仅讨论软件 CMM 及项目管理方面的内容，尽管 CMM 还涉及组织管理和过程管理方面的问题。本书不再讨论评价过程，我所著的《CMM 实践》对它进行了简单描述[8]，而有关评价过程的详细描述参见 S.Masers 所著的《基于 CMM 的内部过程改进评估》（CMM Based Appraisal for Internal Process Improvement）。[9]

1.2.1 CMM 综述

CMM 的目标之一是对成熟的过程和不成熟的（或者特设的）过程加以区别。不成熟的软件过程隐含着在没有很多指导原则的情况下执行项目开发，而项目的结果极大地依赖于团队和项目领导的能力。与之相反，在成熟的软件过程下，项目的执行遵循已定义的过程。在这种情况下，项目的结果不再过分地依赖于人员，而更加依赖于过程。由此可以得出，过程越成熟，结果就越可预测，而项目也更好控制。

使用一个过程执行项目时能够从中期望得到的结果范围称为项目的过程能力

（process capability）。使用一个过程执行项目时所实现的实际结果称为项目的过程绩效（process performance）。显然，过程绩效依赖于过程能力。若要始终如一地改进项目的过程绩效，就必须增强过程能力；而过程本身必须变得更加成熟。

在通往更高成熟度的途中会经历一些良好定义的稳定状态，CMM 把它们称为成熟度等级（maturity levels）。每个成熟度等级规定过程的某些特征，成熟度等级越高就拥有更多在更成熟的软件过程中具有的高级特征。因此，CMM 框架描述了不同成熟度等级的软件过程具有的关键要素。当然，它还规定了软件过程从不成熟的过程变成非常成熟的过程应遵循的路线。这一路线涉及 5 个成熟度等级，如图 1.1 所示。[7]

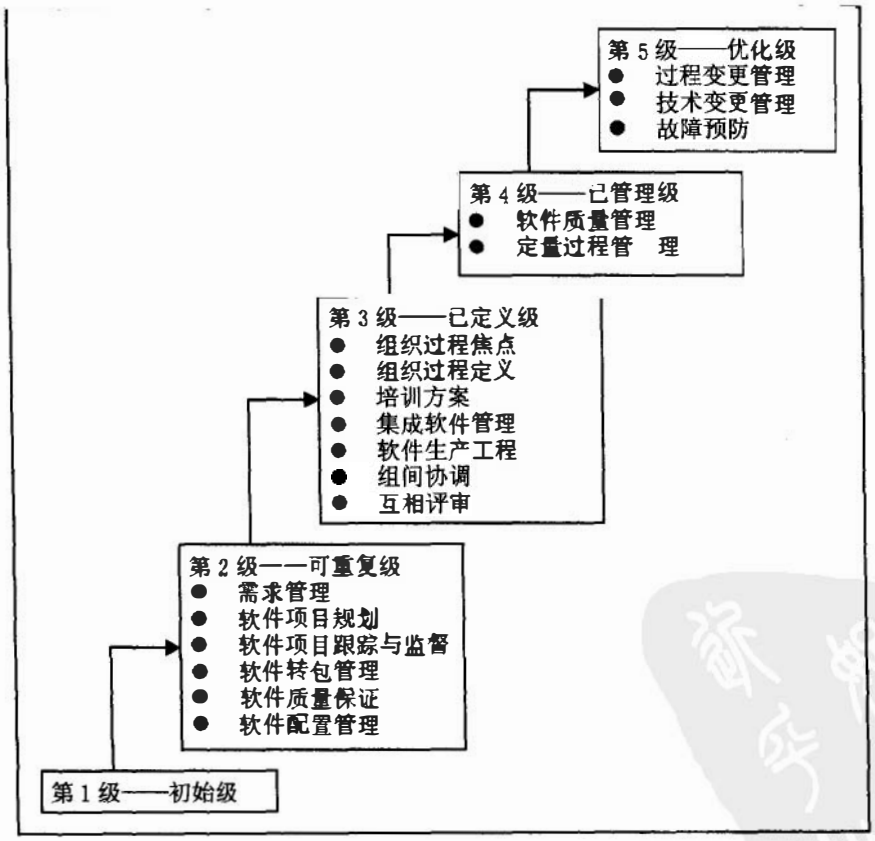


图 1.1　CMM 模型的能力成熟度等级

在第 1 级，即**初始级**，项目按团队和项目经理认为合适的方式进行开发；在

可重复级（第 2 级），虽然还不存在组织范围的过程，但是利用了既定的项目管理实践；在**已定义级**（第 3 级），定义了组织层面的过程，并且得以正确的执行；在**已管理级**（第 4 级），过程能力的量化表示使得有可能定量地预测和控制一个项目的过程绩效；在**优化级**（第 5 级），过程能力以一种可控的方式在改进，并以定量的方式评估这种改进。

每个成熟度等级（除了第 1 级外）都由一些**关键过程领域**（key process area，简称 KPA）所表征，它们规定了组织在将过程提升到那个成熟度等级的过程中应重点考虑的领域。图 1.1 还给出了不同成熟度等级的 KPA。若一个组织要实现某个成熟度等级，它必须实现那个成熟度等级的所有 KPA 以及所有更低成熟度等级的 KPA。

在较高的成熟度等级上维持一个过程是一个具挑战性的任务，需要组织的支持和一个合适的工作文化。在 1996 年和 2000 年 6 月间进行的 900 次评估中（评估结果提交给 SEI），只有 3%的组织达到了第 5 级，5%的组织达到了第 4 级[10]，其余的组织都处于第 3 级或者第 3 级以下，处于第 2 级的组织占 38%，而处于第 3 级的组织占 18%。

1.2.2 项目管理与 KPA

每个 KPA 规定了组织为了满足那个 KPA 必须实现的目标。此外，KPA 规定了一组任务，称为**关键实践**（key practice），由它们共同实现 KPA 的目标。在某种意义上，每个 KPA 的目标表明了其本质。它们规定了 CMM 为那些与 KPA 相关的过程所设的目标。为了说明与项目管理相关的 KPA，我们在此简要地讨论了这些 KPA 的目标。这些目标取自 CMM，只是在某些目标的措辞上略微有些变化。

表 1.1 列出了第 2 级上 KPA 的所有目标，明确地表明了第 2 级的重点几乎全部在项目管理上。在这些目标指导下，你建立一个项目计划并用文档记录它，根据计划评估正在进行的项目的绩效，并在实际绩效明显地偏离计划时采取相应的措施。需求用文档正确地记录下来，并对需求的变更进行正确的管理。所有工作产品都在控制之中，并根据一个事先制定好的配置管理计划正确地管理对产品的变更。执行评审与审计，确保遵循计划的过程和标准。如果项目的某些部分被转包给其他软件开发商，则也要对被转包的工作进行正确地监督。

表 1.1　第 2 级（可重复等级）KPA 的目标

KPA	目标
需求管理（Requirement Management，简称 RM）	● 控制软件需求以建立软件工程和管理任务的基准 ● 保持软件计划、产品和任务与需求相一致
软件项目规划（Software Project Planning，简称 SPP）	● 用文档记录估计，并在项目规划和跟踪时使用 ● 规划并用文档记录项目任务和责任 ● 相关小组和个人在他们应承担的项目责任上取得一致意见
软件项目跟踪和监督（Software Project Tracking and Oversight，简称 SPTC）	● 根据计划跟踪实际的结果和绩效 ● 当实际的结果和绩效显著地偏离软件计划时，采取正确的行动，设法消除偏差 ● 对责任的变更必须得到相关小组和个人的认可
软件转包管理（Software Subtract Management，简称 SSM）	● 主承包者与次承包者在他们应承担的任务上取得一致意见 ● 主承包者根据其责任跟踪次承包者的实际结果 ● 主承包者和次承包者保持联系 ● 主承包者根据其责任跟踪次承包者的实际绩效
软件质量保证（Software Quality Assurance，简称 SQA）	● 规划软件质量保证任务 ● 客观地验证软件产品和任务是否坚持合适的标准、过程和需求 ● 通知相关小组和个人关于软件质量保证任务和结果 ● 由高级管理层解决项目中不能解决的不相容问题
软件配置管理（Software Configuration Management，简称 SCM）	● 规划软件配置管理任务 ● 标识、控制和应用所选的软件工作产品 ● 控制对所标识的软件工作产品的变更 ● 通知相关小组和个人关于软件基准的情况和内容

表 1.2 详细描述了第 3 级上 7 个 KPA 中的 3 个 KPA 的目标。其他 KPA 重点强调组织管理和过程管理问题。对于达到第 3 级能力成熟度的组织中的项目，它使用专用型的标准过程，并重用以往项目的资源、数据和经验进行规划。执行项目的各小组通过良好定义的接口和机制进行愉快地合作。正确地执行评审以标识出工作产品中存在的缺陷，并为执行评审和随后的任务提供充分的支持。

表 1.2　第 3 级（已定义级）中 3 个 KPA 的目标

KPA	目标
集成软件管理 (Integrated Software Management,简称 ISM)	● 项目已定义的软件过程是一种专用型的标准软件过程 ● 根据项目已定义的软件过程规划和管理项目
组间协调（Intergroup Coordination，简称 IC）	● 所有相关的小组都同意客户的需求 ● 所有小组都同意不同小组间应承担的义务 ● 小组标识、跟踪和解决组间问题
互相评审（Peer Review，简称 PR）	● 规划互相评审任务 ● 标识软件工作产品中的缺陷，并消除它们

表 1.3 给出了第 4 级上 2 个 KPA 的目标。在第 4 级上，组织的过程能力用量化术语表示。过程能力用于设定一个项目的量化目标。有关项目绩效方面的数据在当前的基础之上进行收集并与以往的绩效进行比较，如果觉察到明显的偏差，则采取正确的行动恢复对项目的控制。第 4 级的一个关键特点是在当前基础上使用统计过程控制技术，以便在需要时可以评估每种任务并采取正确的行动。

表 1.3 第 4 级（已管理级）上 KPA 的目标

KPA	目标
量化过程管理（Quantitative Process Management，简称 QPM）	● 规划量化过程管理任务 ● 定量地控制项目定义的软件过程的过程绩效 ● 组织的标准软件过程的过程能力用量化的术语进行衡量
软件质量管理（Software Quality Management，简称 SQM）	● 规划项目的软件质量管理任务 ● 定义软件产品质量的可测目标以及它们的等级 ● 量化实现软件产品的质量目标的实际进程，并对它进行管理

第 5 级上的 3 个 KPA 重点强调过程能力的改进。在这 3 个 KPA 中，故障预防 KPA 对项目管理影响最大。这个 KPA 要求通过系统地分析故障的原因，然后消除那些原因，从而前瞻性地预防故障。如果能够预防故障进入软件，则可以减去排除它们的工作量，从而提高质量和生产率。

1.3 Infosys 公司的项目管理

Infosys 公司每年要执行上百个项目。执行一个项目的全部责任由项目经理负责，他必须确保项目团队在规定的成本内按时将高质量的软件交付给客户。为了帮助项目经理实现这一责任，必须得到组织的支持。本节简要地介绍了 Infosys 公司的背景知识以及它对项目管理的支持。

1.3.1 Infosys 公司的背景知识

Infosys 公司是一个软件公司，总部位于印度的班格洛尔。它的宗旨是“成为全球著名的公司，提供由一流的专家制定的一流的软件解决方案”。它利用全球提交模型，客户可以位于世界上的每个角落，而客户所需的解决方案也可以来自世界各地。在这个模型中，客户源自能够为公司提供最大价值的世界各地。对于客户解决方案，综合利用过程、技术和管理对工作进行划分，以便在最佳的位置增

加价值，然后对它们重新聚合后交给客户。

Infosys公司目前有10 000名员工，大约有15个开发中心，分别位于4个国家，并在十几个国家中设有办公地点。公司于1981年由7位软件专家所创立，当时的资产净值只有300美元。今天，Infosys公司的市场资本已经超过80亿美元（基于2001年6月的市场汇率），2000年的收入超过了4亿美元（1994年的收入仅为950万美元）。它的客户遍布世界各地，包括一些大公司——其中60家公司在全球财富排行榜上名列前1000家；它们从事各种各样的业务，诸如银行业、零售业、制造业、电信业、金融服务业、保险业以及交通运输业等。

Infosys公司有着很高的声望，是印度国内管理得最好、最有声望的公司，也是亚洲处于领导地位的信息技术（IT）公司。它获得过很多奖项，包括Ramakrishna Bajaj奖（相当于Malcolm Baldridge奖）。可以毫不夸张地说，Infosys公司是全球最好的软件业务公司之一。

Infosys提供了第一流的基础设施，使项目经理能够更好地服务于世界各地的客户需要。公司为每个小组基本上都配备了音频会议设备，以便项目经理能够轻松地与位于不同地点的客户和小组成员进行交流。同样，公司的各个地点之间的交互用上了视频会议设备，并用它召开虚拟会议。现在，它在班格洛尔的主要阵营已经成为世界上最大的软件业务机构之一，拥有大量与工作有关的设施，诸如图书馆、扩展计算和联网设备、培训设备、讨论室、投影工具等等；还有一些娱乐设施，诸如画廊、健身俱乐部以及网球场和篮球场等。

面向过程和过程改进是Infosys公司的工作文化不可分割的一部分，并且大多数经常执行的任务都定义了过程。对于过程定义和过程改进，Infosys首先采纳了ISO 9000框架，并于1993年获得ISO认证。为了进一步改进软件过程，Infosys采纳了CMM框架。在1997年12月进行的第一次能力成熟度评估中达到了第4级，并在1999年12月达到了第5级。Infosys公司不断追求进取，现在又采用Malcolm Baldridge框架来全面改进所有操作领域，并在所有这些领域中建立领先优势。

1.3.2 SEPG对项目的支持

Infosys公司的质量部门有一个软件工程过程小组（software engineering process group，简称SEPG）。SEPG负责协调所有过程任务，包括过程定义、过程改进以及过程部署。它还管理所有与过程使用相关的信息和数据（诸如过程数据

库和过程能力基准等，这些将在第 2 章做进一步讨论)。

虽然提交产品的全部责任（包括质量）属于项目团队，但是 SEPG 使得项目团队能够轻松地遵循正确的过程。SEPG 还形成了一个独立地监督过程问题和质量问题的渠道，并通过这个渠道向高级管理层报告这些问题。因为“过程不能依靠过程本身得到坚持”，所以 SEPG 有助于确保已定义的过程得以实现并成为标准的实践。

为此，SEPG 除了提供过程培训外，还提供了一个与项目相关的成员，称为**软件质量顾问**（software quality advisor)。质量顾问帮助定义过程和遵循过程，确保过程得以遵守，帮助分析数据，并提供任何所需的过程培训。因为顾问非常精通于项目过程、指南等等，所以顾问的主要帮助在对项目规划的过程中，顾问还评审项目计划，以确保它包含了全部的关键要素。

除了提供咨询并用过程和度量标准进行帮助外，Infosys 公司的 SPEG 还要制定定期独立审计的计划，并对它们进行管理（参见第 11 章)，以确保已定义的过程和标准得以遵守。

1.3.3 高级管理层参与项目

Infosys 以向客户提供价值为荣，这主要通过交付优质的产品得以实现。在 Infosys 公司的任何东西，包括它的组织结构，都是以快速高效率地服务客户和及时把握新的商业机遇为目标驱动的。

在提供客户服务方面，Infosys 公司有很多**业务部门**。在一个业务部门中，一个团队执行一个项目，并由**项目经理负责**。项目经理负责项目执行的方方面面，包括从确定软件的需求到最后的安装。项目经理向**业务经理**负责，而业务经理一般向**业务部门主管**负责。

为了处理项目经理不能解决的情况，高级管理层参与项目是有必要的。在 Infosys 公司，业务经理定期与项目经理进行交流，并通过状态报告和里程碑报告对项目进行监督（参见第 11 章的讨论)。除了定期监督外，业务经理还要帮助解决项目团队不能处理的各种问题，并将这些问题上交（escalated）到他这里（有关问题提交将在第 8 章中讨论)。业务经理还要与客户进行交互，确保他们感到满意，使他们即时地提出各种问题并解决之。

此外，其他高级管理人员也要定期对项目进行评审，他们可以定期参与内部

审计工作（有关审计的问题将在第 11 章中讨论）。通过两个系统——一称为 PRISM（project review by senior management，高级项目评审系统）和 IPM（integrated project management，项目集成管理系统），高级管理人员就可以评审里程碑报告（milestone report）和项目计划。所有高级管理人员都必须通过这个系统定期地评审某些项目，并将有关的注意事项反馈给项目领导。

总之，高级管理人员主要通过监督的方式参与项目，确保项目目标得以实现，并且客户完全感到满意。

1.3.4　项目经理培训

因为项目经理的主要责任是满足客户的需求，所以他们不仅需要掌握项目执行的技术，而且还要与客户进行交互、引出需求、管理团队等等。显然，一个人不可能拥有所需的全部技能，因此关键的做法是对人员进行培训，使他们发展所需的技能。Infosys 已经实施了各种程序来帮助人们从工程师转变成项目领导。

所有新加入的成员都要经历 3 到 4 个月的**入门培训程序**（induction training program）。除了工程和技术方面的培训外，该程序还包含 1 到 2 天在商业礼仪、书面通信、公开演说、手势语等方面的培训计划。

然后，在工程师即将成为**模块领导**（module leader，指那些管理一个系统模块开发的管理人员，尤其是更大项目中的模块）或者项目经理时，他们还要参加一系列技术培训和软技能培训方面的培训计划。前者包含一个 5 天的项目管理课程，重点讲述项目管理的方方面面：规划、监督和控制等等。历时 2 周的需求规范和管理课程教授如何引出需求、如何用文档记录它们、如何验证它们等等。历时 5 天的软技能素质培训方案包括有关估价和团队管理方面的培训、客户关心的问题和客户管理、领导艺术、不同国家的社会和商业礼仪等等。

其他定期提供的课程重点介绍管理方面的知识；项目领导在其时间允许的情况下参与这些课程。此外，团队建设专题研讨会由专业人士主持。

1.3.5　项目管理过程

项目团队要成功地执行一个项目，就必须执行上百个任务，其中有许多任务是互相关联的。有效地管理这一过程对于项目的成功非常重要。在 Infosys 公司，

项目经理执行的一系列任务在**项目管理**（project management）过程中加以规定。这是一个非常标准化的过程，有三个主要阶段：

- 项目规划
- 项目执行
- 项目收尾

在**项目规划**阶段，项目经理审阅合同条款并制定一个满足它们的计划。制定严格的项目计划涉及定义一个生命期过程（这是以后要遵守的过程）、估计工作量和进度、准备一个详细的任务进度计划等等。它还涉及规划质量管理和配置管理以及风险管理。在这个阶段，项目经理的主要任务如下：

- 执行项目启动和管理任务
- 制定一个包含进度的项目计划
 - ✧ 定义项目目标
 - ✧ 标识适于项目执行的标准过程
 - ✧ 裁制标准过程以满足项目需求
 - ✧ 定义一个需求变更管理的过程
 - ✧ 估计工作量
 - ✧ 规划人力资源和团队组织
 - ✧ 定义项目的里程碑并制定一个进度表
 - ✧ 定义质量目标以及实现它们的质量计划
 - ✧ 制定故障预防计划
 - ✧ 标识风险并制定缓和风险的计划
 - ✧ 定义项目的考评计划
 - ✧ 定义项目的培训计划
 - ✧ 定义项目跟踪程序
- 执行项目计划和进度计划的评审
- 获得高级管理人员的授权
- 定义和评审配置管理计划
- 使项目团队适应项目管理计划

除了项目经理外，该阶段还会牵涉到客户、一个 SEPG 代表以及项目的业务经理。该阶段开始的标志是签定了项目合同或者得到了项目授权。而收尾的标志是已经用文档记录了项目计划，并且小组对它们进行了评审（参见第 10 章）。

第 2 阶段是**项目执行**阶段，包括执行项目计划、跟踪项目的状态，并在项目

的绩效偏离项目计划设定的绩效时采取措施进行纠正。换句话说，它将牵涉到项目过程的跟踪和控制。这是项目管理过程中历时最长的一个阶段，它将定期采取一些任务，诸如监督项目状态和质量，以及采取任何所需的纠正措施。在这个阶段，项目经理主要从事如下一些任务：

- 根据项目计划执行项目
- 跟踪项目状态
- 与高级管理层一起评审项目状态
- 监督与已定义的项目过程相吻合
- 分析故障并执行故障预防任务
- 监督程序层面的绩效
- 执行里程碑评审，如果需要重新制定计划

在这一阶段团队的其他成员也要参与。项目开始的标志是项目计划已经完成并得到了批准，而结尾的标志是所有交付的工作产品为客户所接受。

项目管理过程的最后一个阶段是**项目收尾**，主要是在客户接受工作产品之后对项目进行系统的总结。这里的主要目标是从经验中进行学习，以便能够改进过程。项目完成后的数据分析构成了这一阶段的主要任务；对度量标准进行了分析，过程资源（诸如模板和指南等材料可用于帮助管理过程本身）被收集起来供以后使用，并记录有关的经验教训。因为从项目中学习经验是这一阶段的主要目标，所以这是一种群组活动，牵涉到项目经理、SEPG 以及团队中的其他成员。这一阶段开始的标志是客户已经接受了工作产品，而收尾标志是举行了项目完成后的会议。这一阶段的主要结果是项目收尾报告和收集的过程资源。

本书的其余部分讨论这一管理过程的各种要素。第 1 部分主要讨论了关键的规划任务，诸如过程定义和裁制、风险管理、工作量和进度估计、质量规划和配置管理规划。而规划阶段的其他任务（诸如人力资源规划、项目组织、要使用的工具、项目跟踪步骤等等）在第 7 章和第 8 章进行了简单的讨论。第 2 部分主要讨论项目监督、控制以及项目收尾。

1.4　ACIC 案例研究综述

ACIC 公司（为了保密，这里采用原公司的化名）是一个拥有几十亿美元的金融机构。为了与时代同步，它于几年前开始了基于 Web 的应用，并希望实现网

上开户和账户跟踪业务。因为在早些时候，Infosys 公司在一个称为 Synergy（化名）的项目中已经成功地为 ACIC 建立了某些电子业务，所以 ACIC 聘请 Infosys 来分析问题。该工作以时间加材料（time and material，简称 T&M）合同模式进行执行，即客户根据 Infosys 在分析时所做的工作量进行支付。基于分析结果，Infosys 公司成功地竞标得到了该 Web 项目，从而开始了贯穿本书始末的 ACIC 案例研究。该项目按时成功地推出了新业务，并且软件在运行中从未出现过任何问题。本案例研究不同于我在前一本书中所讨论的 WAR 项目案例研究。[6]

ACIC 项目说明了 Infosys 公司在执行一个项目时必须承担的各种项目规划和监督任务。许多与 ACIC 项目管理相关的结果在相关的章节中进行讨论。这些结果包括：

- Synergy 项目的数据，ACIC 项目经理在进行规划时使用（第 2 章）
- 项目的过程计划（第 3 章）
- 需求变更申请的影响分析（第 3 章）
- 工作量估计和高水平的进度计划，以及如何对它们进行描述（第 4 章）
- 质量计划，包含质量目标和实现它们的计划，包括故障预防和评审的计划（第5 章）
- 风险管理计划，描述主要风险、它们的危险程度和影响、它们的优先权，以及高优先级风险的缓和计划（第 6 章）
- 度量和跟踪计划（第 7 章）
- 完整的项目管理计划，包括团队管理计划与客户沟通及提交计划（第 8 章）
- 项目跟踪文档，包括故障日志、问题日志、状态报告和里程碑报告（第 11 章）
- 故障预防的细节，包括故障分析结果以及故障预防计划对项目的影响（第 11 章）
- 完整的项目收尾报告，包括关于质量、生产率、质量成本、故障排除效率等度量标准数据（第 12 章）

1.5 小　　结

软件项目管理也许是一个项目的结果中最重要的因素。如果没有正确的项目管理，项目几乎总会以失败告终。许多组织已经开发了有效的项目管理过程。本

书描述了这样一个组织——Infosys 公司的有效的项目管理过程，Infosys 公司已经达到了 CMM 模型的第 5 级，并且其项目经理已经成功地执行了上百个项目。

本章主要介绍了如下要点：

- 不能孤立地看待项目管理的各个过程，在一个平衡的过程中，可以顺利地将实践汇集在一起。
- 一个组织的过程应当体现它的最佳实践，以帮助新项目效仿以往的成功经验，从而避免失败。
- 在最高级上，项目管理过程包括三个阶段：规划、执行和完成。
- 为了有效地执行项目，项目经理在执行过程时应得到 SEPG 的帮助、高级管理层对问题的监督和问题解决，以及良好的培训。
- 软件 CMM 模型的所有成熟度等级上的许多关键过程领域直接联系项目管理。

1.6　参考文献

[1] L. H. Putnam and W. Myers. *Industrial Strength Software: Effective Management Using Measurement*. IEEE Computer Society Press, 1997.

[2] R. L. Glass. *Software Runaways: Lessons Learned from Massive Software Project Failures*. Prentice Hall PTR, 1998.

[3] International Standards Organization. *ISO 9001, Quality Systems — Model for Quality Assurance in Design / Development, Production, Installation, and Services*. 1987.

[4] International Standards Organization. *ISO 9000-3, Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software*. 1991.

[5] U. K. Dept. of Trade and Industry and British Computer Society. *TickIT: A Guide to Software Quality Management System Construction and Certification Using EN29001*. 1992.

[6] W. Humphrey. *Managing the Software Process*. Addison-Wesley, 1989.（"SEI 软件工程丛书"之《软件过程管理》，瓦茨·汉弗莱著，其英文影印版由清华大学出版社于 2002 年出版，中译版由清华大学出版社于 2003 年出版。）

[7] Carnegie Mellon University/Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, 1995.

[8] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys.* Addison-Wesley, 2000.

[9] S. Masters. *CMM Based Appraisal for Internal Process Improvement（CBAIPI): Method Description. Technical Report, Software Engineering Institute, CMU/SEI-96-TR-007,* 1996.

[10] Software Engineering Institute. Maturity Profile Report, http://www.sei.cmu.edu/activities/sema/profile.html

第 1 部分　项目规划

第 2 章　项目规划基础结构

在迪内森（Dinesh）担任 Infosys 公司的客户发布中心主管期间，当他结束第 10 次项目计划评审会议时，他感到很不高兴。在评审了一个又一个项目计划之后，迪内森注意到每个项目经理都在他自己的领域内努力工作，力争建立最优的项目过程来执行其项目和产生他能够实现的估计——尽管迪内森知道其他团队在早些时候已经执行过类似的项目，并且他们的经验和数据能够极大地减轻项目经理的痛苦。不仅项目经理从头开始重新投资他们的规划工作，而且他们“计划”重蹈他们之前的项目经理犯过的同样的错误。迪内森意识到解决这种困境的办法在于：建立一个所有项目的管理人员都能够访问的公共知识库（institutional memory）。

问题的重点在于如何建立这么一个公共知识库，并用它塑造项目规划的基础结构。该知识库的要素是什么呢？如何系统地记录以往的经验并使它们可重用呢？Infosys 公司又如何保持项目规划基础结构最新呢？

本章讨论 Infosys 公司的规划基础结构的关键要素：过程数据库、过程能力基准和过程资源。在后文中，你将会看到如何在项目规划中使用这些要素。

过程数据库（process database，简称 PDB）保存已完成项目的绩效数据。过程能力基准（process capability baseline，简称 PCB）概括了各个项目的绩效，定量地规定了遵循过程所能达到的结果范围。因此，如果遵循相同的过程，就可以预测出项目的结果范围。过程资源（process asset）是一些文档，诸如检查表、模板、方法以及所吸取的经验教训——即总结以往经验的材料，它们能帮助项目经理和工程师有效地使用过程。

在高成熟度的组织中，一般都存在上述要素，但是它们在各组织中的使用却不尽相同。[1]

2.1 过程数据库

过程数据库是项目过程绩效数据的永久性存储库，可以用于项目规划、估计、生产率和质量的分析以及其他一些用途。[2]PDB 保存已完成项目的有关数据，每个项目提供一个数据记录。正如你所想象的那样，要填充 PDB，就必须收集数据，对它们进行分析，然后组织成记录项的形式。本节主要讨论 Infosys PDB 中的数据是如何表示的；第 7 章说明了如何收集数据。

2.1.1 PDB 内容

在项目规划期间使用 PDB 中的信息时，项目经理经常发现同类项目的信息非常有用。为了实现相似查询，应当在 PDB 中保存项目的基本信息，诸如项目所用的语言、平台、所用的数据库、所用的工具、项目规模和工作量等等。有了这些信息以后，项目经理就可以搜索和查找所有项目的信息。例如，专门查找一个特定的应用领域，使用某种特定的数据库管理系统（database management system，简称 DBMS）或者语言，或者指定某个特定的平台。

为了帮助项目规划，应当获取关于工作量、故障、进度、风险等方面的数据。如果知道了一个项目所花的总工作量，以及不同阶段的工作量及其分布情况，那么这一数据就可以用于估计新项目的工作量。

因此，Infosys PDB 中保存的数据可以分成如下几类：

- 项目特征
- 项目进度
- 项目工作量
- 项目规模
- 故障

关于项目特征方面的数据包括项目名称、项目经理和模块领导的名字（以便能够联系到他们，进一步了解有关信息或者澄清有关事实）、业务部门（以实现基于业务部门的分析）、所部署的过程（以允许不同过程的独立分析）、应用领域、硬件平台、所用的语言、所用的 DBMS、项目目标的简单陈述、关于项目风险的信息、项目的持续时间以及团队的规模。

项目进度计划数据主要包括项目期望的开工日期和完工日期，以及实际的开工日期和完工日期。关于项目工作量的数据包括最初估计的工作量和实际的总工

作量数据，以及各阶段的实际工作量，诸如项目开始、需求管理、设计、构建、单元测试和其他阶段等实际工作量。第 7 章讨论了如何获取工作量数据。

所开发的软件规模可以根据代码行数（lines of code，简称 LOC）来度量，也可以根据简单程序、中等复杂程序、复杂程序的数量来度量，或者综合应用上述两种度量方法。即使在估计时没有采用功能点（function point），但是通过将最终规模用功能点表示，就可以得到生产率的统一度量指标，这通常可以用已公布的转换表将用 LOC 度量的软件规模转换成功能点而得到。[3]也可以获取以功能点表示的最终系统的规模。

关于项目故障的数据包括在各种故障检测任务中发现的故障数量，以及在不同阶段增加的故障数。因此，你要记录从需求评审、设计评审、代码评审到单元测试及其他阶段中发现的故障数。第 7 章说明了项目如何记录故障数据。

此外，还要记录备注，包括关于估计的备注（例如，用于将程序分类成简单程序、中等复杂程序、复杂程序的指标）和关于风险管理的备注（例如，风险认识在项目执行过程中是如何变化的）。

2.1.2　记录项示例

让我们通过一个例子来看看项目的 PDB 记录项，我们用假名 Synergy 来称呼这个项目。Synergy 项目是案例研究项目（ACIC 项目）的前导应用。案例研究在规划期间将参考这个 PDB 记录项。

4 张主表的数据都已给出（本例使用表示名（expressive name），但在实际数据库中各阶段和质量任务都使用代码）。本例的数据非常完备：然而在其他情况下，数据有可能不尽完备。仍然有用的信息数据是不能丢掉的。[4]因此，这种数据还应保存在 PDB 中。

表 2.1 给出了项目的基本信息，包括开始日期和结束日期（估计的和实际的）、估计的工作量（该表没有给出实际工作量，因为可以根据工作量表计算出实际工作量）、团队最多时的人数、风险信息、所用的工具，以及一些其他内容。此外，该表还保存其他一些信息，例如关于客户的信息。

第 2 张表保存工作量信息。对于过程的不同阶段，其工作量数据包括该阶段的任务所花的工作量和任务完成后返工所花的工作量。之所以保存返工工作量是因为它有助于计算和理解质量成本。表 2.2 给出了 Synergy 项目以人时

（person-hour，个人一小时，简称人时）计的工作量数据。表中也给出了各阶段估计的工作量。（该项目的生命期中各阶段总共所花的工作量为 2 950 人时，而评审阶段花了 223 人时；估计的总工作量为 3 012 人时。）

表 2.1 项目的基本数据

字段名	Sydergy 值
ProcessCategory（过程分类）	开发过程
LifeCycle（生命期）	整个开发过程
BusinessDomain（业务领域）	经纪业/金融业
ProcessTailoringNotes（过程裁制备注）	对影响很大的文档增加小组评审；每个开发人员开发的第一个程序接受小组评审
PeakTeamSize（团队最多人数）	12
ToolsUsed（所用的工具）	文档配置管理 CM 用 VSS 工具，源代码用 VAJ 工具
EstimatedStart（估计开始日期）	2000 年 1 月 20 日
EstimatedFinish（估计结束日期）	2000 年 5 月 5 日
EstimatedEffortHrs（估计工作量）	3 106 小时
EstimationNotes（估计备注）	一种用于估计的方法是“用例点”（use case point）方法
ActualStart（实际开始日期）	2000 年 1 月 20 日
ActualFinish（实际结束日期）	2000 年 5 月 5 日
FirstRisk（第一个风险）	完成客户数据库的链接
SecondRisk（第二个风险）	额外需求
ThirdRisk（第三个风险）	人员流失
RiskNotes（风险备注）	岗位轮换：同意在产品验收后采取增强措施：团队建设任务已完成

表 2.2 工作量数据

阶段	任务工作量	评审工作量	估计的工作量
需求分析	0	0	0
设计	414	32	367
编码	1 147	76	1 182
独立的单元测试	156	74	269
集成测试	251	30	180
验收测试和安装	183	0	175
项目管理	237	8	357
配置管理	30	3	38
项目特有的培训	200	0	218
其他	332	0	226

第 3 张表保存故障信息。它不仅需要描述何时检测到故障，而且还要描述何时引入故障，这些信息都是必不可少的。因此，你必须记录每个引入阶段和检测阶段发现的故障数。检测阶段包括各种评审和测试，而引入阶段包括需求、设计和编码。如果你能够根据故障引入阶段按阶段划分检测到的故障，那么你就可以计算出故障检测阶段的故障排除效率。此信息对于识别有可能加以改进的领域很有用。表 2.3 给出了 Synergy 项目的故障数据。

表 2.3　Synergy 项目的故障数据

	需求评审	设计评审	代码评审	单元测试	系统测试	验收测试
需求	0	0	0	1	1	0
设计		14	3	1	0	0
编码			21	48	17	6

最后一张表保存项目的规模信息。因为一个项目可能使用不同的语言，所以该表可能有多个记录项。规模度量也可以采用多种单位，因此该表还要保存规模的单位。一般情况下，如果规模以 LOC 为单位给出，那么也可以根据需要用转换表计算出以功能点为单位的规模。此信息用于根据功能点计算生产率。项目规模是决定生产率的关键因素，但是其他因素（诸如采用的操作系统和硬件）也要保存起来。表 2.4 给出了 Synergy 项目的规模数据。

表 2.4　Synergy 项目的规模数据

LangCode (语言代码)	OSCode (操作系统代码)	DBMSCode (数据库管理系统代码)	HWCode (硬件代码)	MeasureCode (度量代码)	ActualCodeSize (实际代码规模)
Java	Windows	——	PC	LOC	8 082
Persistent Builder	WindowsNT	DB2	ClientMC	LOC	12 185

2.2　过程能力基准

PDB 保存每个项目的有关数据，而过程能力基准表示在某些时间点上过程能力的量化瞬态图。实质上，过程的能力是指在遵循过程的情况下可对项目期望的结果范围。[5]一个稳定的过程的能力可以由过程以往的绩效所决定。如果按规定建立了基准，则可以轻松地获得过程能力的趋势——这就是需要有一个 PCB 的主要原因。

首先必须解决的问题是 PCB 应当包含哪些内容，即 PCB 应当包含什么类型的“结果”。Infosys 公司的 PCB 包含过程绩效，主要用生产率、质量、进度计划以及工作量和故障分布表示。它规定了如下内容：

- 已交付软件的质量（delivered quality）
- 生产率
- 进度计划
- 工作量分布
- 故障引入率
- 过程中故障排除效率
- 质量成本
- 故障分布

此信息可用于项目规划。例如，项目经理可以用生产率和预测的规模来估算项目的工作量，可以用工作量的分布情况预测各阶段的工作量并制定人员分配计划。同样，可以用故障引入率预测总的故障数，用故障分布预测各故障检测任务检测到的故障数。总故障排除效率或者质量可以用来预报软件交付以后可能出现的故障数，以及制定维护计划。

PCB 在组织的总体过程管理中也起到重要的作用。例如，通过分析 PCB 随时间的发展趋势，可以容易地度量过程改进。通过使用工作量和故障分布信息、故障引入率、故障排除效率和其他度量，还可以改善过程改进的激励计划。

因为基准表示一个过程的能力，所以必须为组织中的每个过程建立一个独立的基准。在 Infosys 公司，分别为维护、重建和开发项目定义了不同的过程；因此，为这些过程中的每一个都定义了一个不同的基准。然而，这些过程太广泛，并且只提供了基本指南。如果某种类型的项目经常执行，则应建立那种项目的 PCB。在项目期望的结果方面，这种专用的基准给出了更加严格的结果范围。

表2.5给定的PCB适用于开发项目和用第3代语言（third-generation language，简称 3GL）开发的项目。例如，在维护项目的 PCB 中，不仅 PCB 中的实际数据不同，而且所包含的信息也不同（以适应维护过程）。（由于保密原因，该 PCB 中所列的数据经过了加工处理。）

表 2.5　开发过程的过程能力基准

序号	参数	备注	基本基准，开发项目
1	已交付产品的质量	已交付产品的故障/功能点(FP)(已交付产品的故障=验收故障+保修故障)	0.00～0.094 个已交付产品的故障/FP（平均：0.021）
		用工作量表示质量	0.00～0.012 已交付产品的故障/人时（平均：0.003）
2	生产率	第 3 代语言 第 4 代语言	4～31FP/人月（平均：12） 10～129FP/人月（平均：50）
3	进度计划符合度		81%的项目在原定进度计划的 10%以内交付。
4	工作量		
4.1	构建工作量	中等复杂程序的构建工作量	最小 平均 最大 2　4　6 人日
4.2	工作量分布		最小 平均 最大
		需求分析+设计	1　15　29%
		构建（代码+代码评审+单元测试）	22　41　60%
			14　33　52%
		编码	
		评审	1　4　11%
		单元测试	1　5　14%
		集成测试+系统测试	1　9　20%
		验收测试与保修	1　8　23%
		项目管理+配置管理	1　10　20%
		培训	1　7　14%
		其他	1　10　26%
5	故障		
5.1	故障引入率	总的生命期故障引入率（以规模为单位）	0.02～1.12 个故障/FP（平均：0.33）
		总的生命期故障引入率（以工作量为单位）	0.00～0.1516 个故障/个人工作小时（平均：0.052）
		编码阶段的故障引入率（以工作量为单位）	0.02～0.57 个故障/个人工作时（平均：0.155）
5.2	故障引入分布	需求和设计	大约 30%
		编码	大约 70%
5.3	过程中故障排除效率	适用于整个生命期	78～100%（平均：94%）
6	质量成本	（评审工作量+返工工作量+测试工作量+培训工作量）占工作量的百分比	32%
7	故障检测分布	占总故障数的百分比	最小 平均 最大
		需求规范评审+概要设计(HLD)评审+细节设计评审	2　13　20%
		代码评审+单元测试	21　53　83%
		集成测试+系统测试	3　28　56%
		验收测试	1　6　17%

该 PCB 的解释比较简单。该 PCB 指出，开发项目的平均生产率大约 12 个功能点/人月，最少为 4 个功能点/人月，而最多为 31 个功能点/人月。这些项目的平均质量是 0.02 个故障/功能点，最小为 0.00 个故障/功能点，而最大为 0.094 个故障/功能点。PCB 还给出了其他参数的平均值及其范围，诸如故障引入率和总的故障排除效率等。总故障引入率根据规模以及工作量的形式给出，因此工作量以及规模估计可用于估计故障量。质量成本包括预防故障或者排除故障的所有任务所需要的成本。对于工作量、故障和故障引入率，它还规定了各阶段的分布情况。

2.3 过程资源和知识库系统

在第 1 章中已经讨论过，过程涵盖了组织成功地开发过程的经验。然而，过程描述通常包含要执行的步骤序列，标识出谁将执行它们，规定主要步骤的开始标志和结束标志等等。为了简化过程的使用，通常可以使用指南、检查表和模板提供的支持，这些材料总称为过程资源（process assets）。

指南（guidelines）通常给出执行某个步骤的规则和程序。例如，项目规划过程中的一个步骤是“工作量估计”。为了执行这一步骤，项目经理需要参照相应的指南。检查表（checklists）通常有两种类型：任务检查表和评审检查表。顾名思义，任务（activity）检查表就是构成一个过程步骤的任务列表。评审（review）检查表的目的是使评审人员对可能在产品中发现的故障引起注意。模板（templates）实质上提供了一个能够记录过程或者步骤的结果的文档结构。图 2.1 表示了过程与这些资源间的关系。

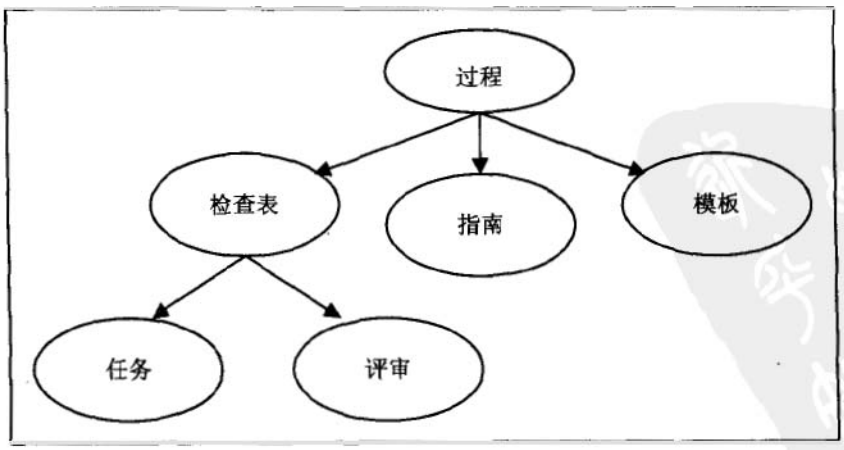


图 2.1 过程资源

这些过程资源的主要目的是简化过程的使用，减少工作量，从而提高生产率。例如，用模板建立文档比从头开始建立文档要更加容易，而且所需的时间也更少。这些资源也有助于改进项目的质量，首先通过提供正确的指南和任务检查表，使引入的故障数最少；然后通过辅助评审，从而更加容易地捕获引入的故障。

简而言之，为了使项目执行从面向过程的方法中取得最大的效益，关键是保存并使用过程资源。在 Infosys 公司，所有的指南、检查表和模板都放在网上共享，并定期对它们进行更新。表 2.6列举了这些材料与项目管理的关系。

表 2.6　用于项目管理的指南、检查表和模板

指南	检查表	模板/表单
工作量和进度估计指南	需求分析检查表	需求规范文档
小组评审程序	单元测试和系统测试计划检查表	单元测试计划文档
过程裁制指南	配置管理检查表	验收测试计划文档
故障估计和监督指南	状态报告检查表	项目管理计划
度量和数据分析指南	需求评审检查表	配置管理计划
风险管理指南	功能设计评审检查表	度量标准分析报告
需求可跟踪性指南	项目计划评审检查表	里程碑（milestone）状态报告
故障预防指南	C++代码评审检查表	故障预防分析报告

除了这些通用的资源（Infosys 质量管理系统的一部分）外，项目经理可能希望重用以往某个在某些特征上比较相似的项目的一些结果。结果重用可以减少工作量和提高生产率。为了促成这一目标，也可以在项目收尾时收集项目的过程结果。通常所收集的可以在不同的系统上使用的资源包括：

- 项目管理计划
- 配置管理计划
- 进度计划
- 标准、检查表、指南、模板和其他有用的东西
- 开发的工具及有关的备注
- 培训材料
- 其他可以为以后的项目重用的文档

虽然过程资源试图通过检查表、模板等来概括经验，但是它们并不能完全获取执行项目时得到的各种形式的知识。获取和重用不同形式的知识需要正确的知识管理，这在基于知识的组织（诸如方案提供商和咨询公司等）中显得越来越重要。许多组织开发了相应的系统，以此有效地利用其雇员们集体的经验和知识。

在 Infosys 公司，除了过程数据库和过程资源外，还用一个称为知识库（body of knowledge，简称 BOK）的系统来总结经验。

基于 Web 的 BOK 系统有它自己的关键字，即基于作者的搜索工具。BOK 中的知识主要以文章的形式出现，并按主题进行组织。基本主题包括：

- 计算机与通信服务
- 需求规范
- 构建
- 工具
- 方法和技术
- 教育与研究
- 设计
- 评审、检查和测试
- 质量保证和生产率
- 项目管理

BOK 系统包含与所吸取的教训和最佳实践相关联的有价值的文章。记录项是通用的，并且不专属于某个特定的项目。基于一个为此目的而建的模板，组织的任何成员都可以递交一个记录项并把它保存到 BOK 中。每次递交的记录项都要接受评审，主要考虑它的有用性、通用性、所需的变更和其他特征。而编辑控制是为了确保记录项满足质量标准。对于向 BOK 递交信息的雇员将得到经济上的奖励，同时管理 BOK 的部门也在积极地寻求新文章。为了进一步推动这一事业，Infosys 公司将向 BOK 递交信息作为员工每年业绩评定的一个因素。

2.4 小　　结

如果存在以往同类项目的经验，则项目经理能够更好地对项目进行规划。项目规划基础结构可以帮助项目经理有效地获取数据和所吸取的经验教训，并使它们可以为项目经理所用。Infosys 公司规划基础结构的主要组成部分有过程数据库、过程能力基准以及过程资源。这些要素具有如下关键特征：

- 过程数据库保存已完成项目的绩效数据。它包含风险数据、工作量和工作量分布、故障和故障分布、项目规模以及其他项目特征。
- 过程能力基准概括了各种过程的绩效，从而规定了在遵循过程的情况下

可对项目期望的结果范围。它包含诸如质量、生产率、故障排除效率以及工作量和故障分布等度量。

- 过程资源是文档，诸如检查表、模板、方法和指南。它们通过减少执行某些任务所需的工作量来提高生产率，通过减少故障或者尽早捕获故障而提高质量。

如果组织没有这些要素，则项目经理可以设法建立由其中的某些要素组成有限的项目规划基础结构。几个项目经理聚在一起，将他们的经验汇集起来就可以建立一个有限的 PDB。这里的关键问题是进行项目后评价（这将在第 12 章中进行讨论）。在你建立了一个有限的 PDB 之后，你还可以抽取一个 PCB 的某些特征。建立一个过程资源集更加容易：你只要收集模板、检查表以及项目中所用的类似材料就可以了。然后，你就可以对它们进行评审和加工，使它们可用。

这些基础结构要素不能直接满足与项目管理相关的 CMM KPA 的需求。然而，它们是满足 CMM-3 和 CMM-4 中许多与项目管理相关的 KPA 所必须的。本章讨论的体系也满足 CMM 的其他一些 KPA。例如，过程数据库是 CMM-3 显式要求的，它对过程资源进行管理。过程能力基准是 CMM-4 的两种 KPA 所需的。

2.5　参考文献

[1] P. Jolote. *Use of Metrics in High Maturity Organizations*. SEPG2K Conference, Seattle, 2000.

[2] W. Humphrey. *Managing the Software Process*. Addison-Wesley, 1989.（"SEI 软件工程丛书"之《软件过程管理》，瓦茨·汉弗莱著，其英文影印版由清华大学出版社于 2002 年出版，中译版由清华大学出版社于 2003 年出版。）

[3] C. Jones. *Applied Software Measurement: Assuring Productivity and Quality*, second edition. McGraw Hill, 1996.

[4] R. Grady and D. Caswell. *Software Metrics: Establishing a Company-wide Program*. Prentice Hall, 1987.

[5] Carnegie Mellon University/Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, 1995.

第3章 过 程 规 划

与许多项目经理一样，拉维（Ravi）将软件开发的瀑布模型作为主要的软件生命期过程加以研究。他已准备好用它来开发即将从事的项目，这是他的第一个项目。然而，拉维发现不能采用瀑布模型，因为客户希望按阶段交付软件，这就隐含着必须按部分交付和建立系统，而不是作为一个整体。

在许多其他项目中也会出现与此非常雷同的情况。在实际情况中，几乎不存在某个标准过程或者以往某个项目所用的过程是最适合于当前问题的过程。为了找到最合适的过程，就要根据新问题对现有过程进行裁制。

本章描述Infosys公司所用的开发过程和过程裁制。即使开发过程经过裁制以后，通常也不能处理变更申请。为了满足变更申请而同时又不失去对项目的控制，必须为开发过程增加需求变更管理过程。因此，本章还描述了Infosys公司采用的需求变更管理过程。

3.1 Infosys开发过程

在项目规划期间，项目经理必须决定采用什么过程来设计软件。这是一个非常关键的问题，因为大量的工程任务将受此决定支配。这就好比一次漫长的汽车旅行一样，计划的路线决定了驾驶的方向。

软件开发存在几种过程模型。最常见的模型包括瀑布模型（有关该模型的描述及其存在的局限性，参见勃姆（Boehm）的《软件工程经济学（Software Engineering Economics[1]）》）、迭代式增强模型[2]、原型法开发模型[3]和螺旋模型[4]。使用最广泛的模型是瀑布模型，它按线性顺序组织各开发阶段，但是大多数实现都对这个模型进行了改进以使它的缺陷最小。

宏观上，一个标准过程可以给出某类项目各阶段的最优组织，并给出过程定义的良好起点。但是，标准过程并不适用于所有情况，最好的过程可能是某种对标准过程处理后得到的过程。因此，为了决定使用哪种过程，项目经理必须选择基本过程，同时还要决定如何对它进行裁制以得到适合项目的过程。本节后面的

内容讨论 Infosys 公司采用的标准开发过程，并说明项目经理如何对它进行裁制。

3.1.1 标准过程

Infosys 公司采用的标准过程类似于瀑布模型，但是原来的各个阶段被划分成更小的阶段，或者说状态，以允许并行地执行某些阶段。例如，可以将系统测试的规划定义为系统测试本身的一个独立的阶段，这是一种使团队可以并行地执行系统测试规划阶段和编码阶段的实践，尽管系统测试在编码完成后才进行。

该标准过程的各阶段包括需求分析、概要设计、详细设计、构建、单元测试、集成计划、集成、系统测试计划、系统测试、文档、验收和安装、维护支持。图 3.1 描述了上述各阶段以及它们之间的依赖关系。这里就不介绍各阶段的细节了，若想了解有关它们的细节可以参考我以前所著的书。[5]

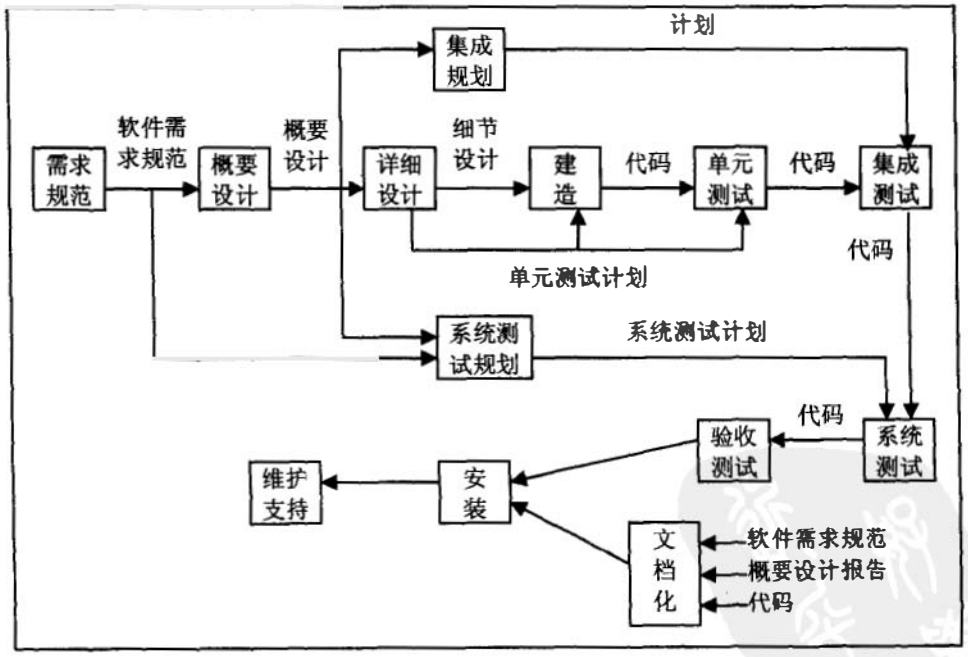


图 3.1 Infosys 公司采用的开发过程

该过程的形式化描述规定开始条件和结束标志、输入和输出、参与者、任务以及各阶段（或者状态）的其他信息。过程描述一般比较简单，规定那个阶段所

采取的任务列表。

即使项目采用面向对象的方法，总体过程仍然相同，只是某些阶段存在不同罢了。不同点主要存在于系统分析与设计阶段，不过某些后期阶段采用的指南和标准也有所不同。

对于进行迭代式开发或者原型法开发或者仅执行生命期中某些阶段的项目，也可以采用该基本过程。在这些情况下，需要对这个基本过程加以调整才能满足项目的需要。这些调整通过过程裁制（process tailoring）完成，下面讨论过程裁制。

3.1.2 过程裁制

任何一个已定义的过程（不管是一个组织的标准过程，还是以往某个项目采用的过程）都不能满足所有的情况和所有的项目。已定义的过程需要经过裁制才能满足当前项目的需要。

裁制是指调整组织以前定义的过程以获得一个适用于某个项目的特定业务或者技术需要的过程。[5,6]你可以将裁制看成添加、删除、修改一个过程中的任务，使得最终的过程更加适于实现项目的目标。

不受控制的裁制实际上隐含着从头开始建立一个过程。为了有效地使用以往定义的过程，有必要提供一些裁制指南。这些指南定义了应当对一个标准过程进行修改的条件和类型。实质上，它们（指南）定义了标准过程允许的偏差（permitted deviation），希望能够为项目定义一个较优的过程。图 3.2说明裁制指南的作用。

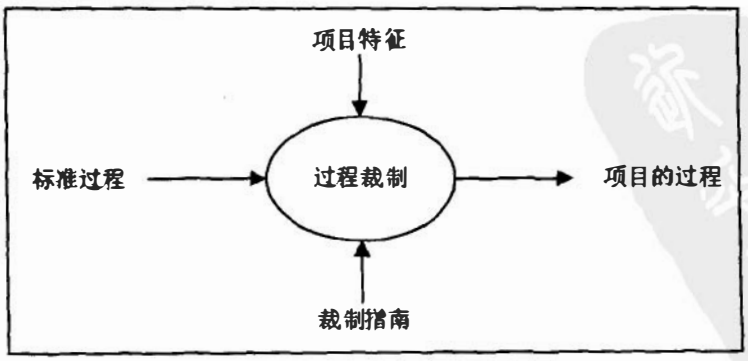


图 3.2 过程裁制

为了说明裁制的需要，让我们看看开发过程构建阶段中的一个任务——代码评审。在许多情况下代码评审会大大地增加价值，但是增加的价值并不与所需的工作量相称。此外，评审既可以由一个小组完成（遵循小组评审程序），也可以由一个人完成。标准开发过程没有规定应当如何执行代码评审。通过只为某种类型的程序（诸如，复杂程序或者外部接口）建议执行代码评审任务和建议较优的评审方式（小组评审，或者个人评审），裁制指南能够为项目经理提供帮助。

Infosys 公司采用的裁制方法类似于金斯伯格（Ginsberg）和奎因（Quinn）提出的基于表的方法[7]，项目经理规定过程要素、可裁制的属性、每个属性的选项以及选择特定选项的原因。项目经理可以在两个层次上执行裁制：概要级裁制和详细级裁制。

概要级裁制（Summary-Level Tailoring） 在概要级裁制，项目经理根据项目特征，应用总体指南对标准过程进行裁制。即，它提供了一些关于某些类型的详细任务的基本规则。执行该步骤时，项目经理首先标识出项目的某些特征。对于开发项目，过程裁制将用到如下特征：

- 团队和项目经理的经验和熟练程度
- 团队人数最多时的人数
- 需求透明度（clarity of the requirement）
- 项目持续时间
- 应用的关键程度

如果团队的大部分成员对项目所采用的技术有 2 年以上的经验，则设定这个团队的经验水平为高；否则为低。如果应用程序对客户的业务或者 Infosys 公司的业务有非常显著的影响，则设定应用的关键程度为高，否则为低。如果项目必须在 3 个月内完成，则认为项目的持续时间非常短。

对于这些特征的不同值，都提供了概要级裁制指南。概要级裁制指南一般与评审、工作量、进度计划、资源及过程形式化有关。与评审有关的裁制指南通常规定什么时候执行评审以及应当执行什么类型的评审。这些基本指南规定了详细级过程裁制的环境，并为项目定义了一个合适的过程。

详细级裁制（Detailed Tailoring） 详细级裁制涉及任务的执行、任务的评审和文档化需要。裁制指南可以规定一个任务是可选任务，如果任务是可选的，则项目经理可以决定是否执行该任务。同样，某些文档的准备也可以是可选的，如果文档是可选的，则项目经理可以决定项目是否需要该文档。对于评审，一般有

小组评审、个人评审和不评审。此外，项目经理也可以增加一些新的任务或者可以重复某些任务。

详细级裁制完成后，项目过程要执行的任务序列就定义好了。然后，这些定义就可以用来规划和调度任务，构成项目执行的基础。因为裁制的执行是项目计划中最重要的任务，所以在评审计划时也要评审过程定义和裁制。

3.1.3 裁制示例：短期项目的裁制

我们通过介绍短期项目的概要级裁制来说明裁制的概念。Infosys 公司注意到软件项目的持续时间在逐年下降，并且对持续时间非常短的项目的需求有所增加。显然，这种项目要求对标准过程进行裁制，以允许最大程度地并行执行和非常严格的项目监督和控制。过程裁制取决于需求的透明度、团队和项目领导的经验水平、团队的规模等等。表 3.1 给出了短期项目的各种特征的取值以及项目经理准备的裁制指南。

因为进度计划的主要特征是短期性，所以与进度有关的指南建议采用所谓的时间盒控制法（timeboxing）技术和微型里程碑。对于时间盒控制法，规划以几周为限的周期，并在每个周期结束后交付一个工作系统。为了严格控制时间盒，还要设定微型里程碑（周期内的里程碑）。在微型里程碑处，只需进行有限的里程碑分析（参见第 11 章）。其他指南大多都是不言而喻的。

如果项目经理必须执行一个短期项目，那么他就可以使用这些指南对开发过程进行裁制。例如，项目经理可以计划在较短的迭代过程中进行迭代式产品交付、执行有限的但是更为频繁的里程碑分析、定义一个变更管理过程（将大多数变更申请延迟到下一次迭代）等等。此外，在执行详细级裁制以定义项目的过程时，项目经理可以减少每个周期的文档，例如，通过减少用于报告状态的模板的范围。

表 3.1 所示的概要级裁制指南适用于持续时间少于 3 个月的项目，团队的最多人数超过 5 人，从事的是新技术，并且其项目领导很有经验。在详细级裁制阶段，项目经理根据概要级裁制指南和对项目的适应度选择方案。然而，如果裁制指南不足以令你选择正确的项目过程，则必须修改裁制指南不支持的过程。这种偏差代表潜在的风险，必须在项目管理计划中加以强调，并且该项目管理计划必须经过评审并得到批准。

表 3.1 短期项目的裁制指南

值	概要级裁制指南	理由
团队人数>=5 持续时间<3个月 需求透明度低 新技术 有经验的项目领导	**与进度计划有关的指南**	
	● 规划微型里程碑（mini-milestones）	能够更早地标识出问题并能更好地控制项目 与进度计划有关的风险减少，因为对进度的监督更加频繁 工作量蔓延（effort slippage）有可能在项目的早期得到控制
	● 尝试用时间盒控制交付的产品	进度计划表有可能被缩短，因为它保证在固定的时间间隔内交付系统和阻止特征变化 与进度计划有关的风险减少
	与工作量有关的指南	
	● 如估计指南中定义的那样，应用相应的调整因子作为估计的基础	在项目估计中要考虑所有的项目约束，以计划一个切实可行的时间线
	● 提供乐观估计、悲观估计和概率估计	减少了与不正确的估计和“匆忙（quick）”承诺有关的风险
	● 在设计阶段结束时修正并重新协商估计	
	与资源分配相关的指南	
	● 为任务分配熟练的人员	生产率将提高
	● 对与绩效有关的问题取得专家的帮助	与进度有关的风险减少 由于绩效调整而造成的返工和推迟被最小化
	与过程形式化有关的指南	
	● 标识出所有的投资者及其目标，并对它们划分等级	有助于强调明确的目标 减少了与进度相关的风险，因为已经标识出重要的成功因子，对它们取得了一致意见，并着手处理它们
	● 定期向高级主管报告风险缓和行动的情况	与进度有关的风险被减少，因为风险缓和工作做在前面
	● 定义形式化的需求变更管理过程，并要求团队严格地遵循该过程	最小化需求变更对进度的影响

3.2 需求变更管理

需求会发生变化，并且需求的变更可以在项目生命期的任何时间发生（甚或在项目生命期结束后发生）。需求变更越是在生命期的后期发生，对项目的影响越严重。除了希望不会发生那样的变更，或者希望初始需求做得非常完善并不需要任何变更外，更好的做法是在出现需求变更时处理变更申请。不受控制的需求变更会对项目的成本、进度和质量产生有害的影响。需求变更最多可以占总成本的40%。

拉维意识到迫切需要一个需求变更管理过程。在努力使客户感到满意时，他欣然同意客户申请的变更。由于没有对变更申请加以控制，因此项目经历的需求变更越来越频繁。最后，团队成员不得不每周工作 60 小时，并且工作量增加 100%以上，拉维才勉强地完成了项目。更糟的是，他发现客户仍然感到不满意，因为他认为变更的越多会使产品更好。

在另一个项目中，玛丽（Mary）是一位经验丰富的项目经理，她记录客户每次通过电话或者 e-mail 发出的变更申请，然后告诉客户有关变更申请对工作量和进度产生的影响。不仅变更申请的频度随时间减少了，而且客户将很多申请转移到下一个版本中去实现。最终结果如何呢？项目按时完成，客户感到很满意，并支付了实现变更申请招致的额外工作量。

本节讨论 Infosys 公司采用的需求变更管理过程。该过程定义了出现新需求或者修改现有需求时执行的任务集（我们将这两者都称为需求变更）。如果项目经理有过拉维和玛丽那样的经历，他们就可以采用需求变更管理过程；将需求变更管理过程规定为项目管理计划的一部分，与其说这是一种规划任务，不如说是一种向客户兜售过程并使他们购买的方法。

3.2.1 变更管理过程

在项目规划期间，项目经理决定遵循哪个过程来处理变更申请。计划的过程要与客户进行讨论，以便客户和厂商在如何管理变更方面取得一致意见。通常，变更管理过程规定如何发出变更申请、何时需要正式批准，等等。在出现需求变更申请时，必须执行需求变更管理过程。

因为变更申请意味着需要额外的工作量，所以必须在额外的工作量支付方面

取得一致意见。通常，在客户认可的情况下，项目必须在实现变更申请方面留有一定的余地（通常占总的项目工作量的小百分比）。这样一种预算条款简化了实现客户认可的变更申请的管理问题。

Infosys 公司常用的变更管理过程包括如下步骤：

（1）记录变更。
（2）分析变更对工作产品的影响。
（3）估计变更申请所需的工作量。
（4）重新估计交付时间表。
（5）执行累积的成本影响分析。
（6）如果影响超出一定的限度，则与高级主管一起评审影响。
（7）客户不再提出变更申请。
（8）修改工作产品。

维护一个变更申请日记，以跟踪变更申请。日志中的每条记录项都包含一个变更申请号码、关于变更的简单描述、变更的影响、变更申请的状态以及关键日期。在评估变更申请的影响时，必须执行影响分析。影响分析涉及标识出需要进行变更的工作产品，并估算对每种产品的变更量；通过重新查看风险管理计划，重新评估项目风险；评估需求变更蕴涵的总的工作量和进度估计的变化。客户对分析结果进行评审，并作出答复。变更申请一般作为需求规格说明文档的附件。有时还要修改文档的有关部分以反映所做的变更。监督已认可的变更申请并保证它们正确实现，这部分由配置管理过程处理，我们将在第 9 章进行讨论。

如果实现变更所需的总工作量不超过某个预先确定的值（例如，2 人日），则认为此变更是次要的。次要变更通常作为项目工作量的一部分，利用计划的估计中预留的缓冲工作量。主要变更通常对工作量和进度有更大的影响，并且必须得到客户的正式批准。高级主管通过状态报告和里程碑报告掌握变更情况，这将在第 11 章中讨论。

3.2.2 示例

为了规定变更和变更管理过程的结果，项目通常采用一个简单的模板。每个变更都被分配一个惟一的参考号（reference number），在模板的申请号（request number）字段中做规定。变更规范（change specification）部分简单描述申请的变更。变更的种类（例如，设计变更、合同变更、功能变更、绩效变更等）和变更

的性质在变更分类（change category）中规定。影响分析的摘要被记录下来，同时还要记录可能受到影响的工作产品的简要信息、牵涉到的工作量以及进度影响。状态（status）部分记录即将着手处理的变更申请的状态，还可以记录变更申请的日期。如果需要批准的话，还要记录变更得到批准的日期。通常，项目自定义该模板以满足它们的需求。

图 3.3 给出了 ACIC 项目的一个已填好的变更申请模板。模板的各组成部分的含义是不言自明的。本例是一个主要变更申请，既影响工作量又影响进度。影响分析的结果说明这些因素的影响。分析报告还表明客户已经接受了影响分析（包括工作量和进度的变更）。

申请号： 3　　**日期：** 2000 年 8 月 11 日

变更说明

本变更申请是使客户机屏幕根据监视器自动地调整分辨率。这一变更是必须的，因为 ACIC 项目的所有监视器的分辨率都是 1600×1200，而其他使用该应用的商业伙伴可能使用分辨率 800×600。

影响分析

变更分类： 主要变更，因为它影响所有屏幕。

解决方案： 布局管理器和每个组件的约束设置必须加以修改。由于 Applet 实现，因此必须变更代码，以便屏幕自动地适应监视器的分辨率。

工作量影响： 总屏幕数为 40。实现一个屏幕的分辨率将用 12 个小时左右。因此，估计实现这一变更所需的总工作量是480 人时（大约 53 人日）。

进度影响： 因为团队的平均人数为 5.5 人，所以实现此变更申请对总进度的影响大约为 10 天。

状态

影响分析得到客户的认可，并立即加入到需求规格文档中。项目进度将发生变化；在需要的地方，里程碑的日期也可能发生变化。

经手人： ×××　　**评审人：** ×××

图 3.3　ACIC 项目的变更申请的影响分析

图 3.4 给出了另一个变更申请的例子。在这个例子中，影响分析的具体内容对于理解需求变更管理并不重要。

XYZ项目

申请号： 11 **日期：** 1998年2月23日

变更说明： IS-41分析器——CDMA的IS-41分析器支持。

影响分析

不必专门对CDMA的配置模块和分析器进行修改。可以重用TDMA代码，脚本也可以重用。netconfig和analyzer类可以被重用。受影响的模块有cgaapp模块、cdmaroi模块和silver06guiapp++模块。cgaapp模块：必须触发IS-41分析。cdmaroi模块：（a）TRIS41ROI必须复制成TRCDMAIS41ROI；（b）在TRCDMAROI中有一个纯虚拟方法用于设置ActualCallModelManager。需要对它重新定义。silver06guiapp++模块：必须在资源表中增加IS-41。

进度影响： 无。

工作量影响： 5。

状态： 将被放入新的CDMA包中。

图3.4 另一个变更申请示例

注意，虽然用模板规定了变更申请的影响，并得到了客户的认可，但是对变更申请实现的实际跟踪却由配置管理过程处理，我们将在第9章讨论配置管理过程。

需求变更的一种危险是：尽管每种变更本身并不大，但是生命期中所有变更的累积影响却是很大的。因此，除了研究每个变更的影响并对它们进行跟踪外，还必须监督变更的累积影响。对于累积变更，可以使用变更日志。为了简化这一分析，通常用电子表格对日志进行维护。表3.2所列的信息说明了如何维护累积改变。对于每个申请的细节，可以用变更申请号和日期访问相关的变更申请。

从这种类型的电子表格中，你可以直接看出目前已完成的需求变更的总成本。前面已经提到过，Infosys公司的项目经理有时为处理变更申请预留一定的余地。只要所有变更累积的工作量不超过这一预留的工作量，就不需要做任何特殊的处理。然而，如果所有变更的累积工作量超过了这一预留的工作量，则再进行变更可能会对总成本和进度安排产生负面影响。在这种情况中，项目经理必须修改估计并使它们得到承认。

表3.2 跟踪变更的累积影响

变更申请号	变更申请日期	变更说明	工作量影响（人日）	状态
1	2 月 18 日	规定使用情况统计	3	已解决，2 月 22 日
2	演示期间	用户分组	2	未解决
3	演示期间	强制用户退出	2	未解决
4	2 月 18 日	知识用户的存档	5	已解决，2 月 27 日
5	演示期间	复制窗口	1	未解决
6	演示期间	保存一棵扩展树，并根据需要检索相同的树	10	未解决
7	演示期间	从一个特定的节点启动的能力	2	未解决
8	演示期间	删除时列出所有节点	1	未解决
9	2 月 18 日	注解（建立/删除/批准/修改等等）	10	未解决
10	2 月 23 日	PFNETCONFIG——压缩格式的 netconfig 支持	10	未解决
11	2 月 23 日	IS-41 分析器——CDMA 的分析器支持	5	已解决，3 月 1 日
		总计	51	

3.3 ACIC 项目的过程规划

因为 ACIC 项目是一个开发项目，所以遵循标准的 Infosys 开发过程。然而，对它进行了裁制以满足 Rational 统一过程（Rational Unified Process，简称 RUP）方法的需求，因为向客户承诺使用 RUP。RUP 是一种迭代式开发方法。Infosys 公司采用的过程裁制指南支持迭代式地执行不同阶段，但要在每次迭代中执行分析、设计、编码和测试任务。除了进行迭代式开发外，为了满足 RUP 方法，需对标准过程做如下主要修改：

- 只有那些在一个特定的迭代中开始使用的用例才在迭代中进行详细描述。
- 在最初几次迭代中，增量式地开发逻辑对象模型和物理对象模型。
- 物理数据库设计可以在以后的迭代中进行细化。
- 每次迭代将开发一个单元测试计划。
- 记录每次迭代过程发现的故障。

此外，不准备用标准的可跟踪性矩阵机制进行需求跟踪。相反，将用 Requisite Pro 工具跟踪需求，该工具是开发环境的一部分。

需求变更管理将使用标准过程。尽管在每次申请时都进行影响分析，但是，如果一个变更申请可能要占总工作量的 2%以上时，则必须对影响进行重新估计。过程规划结果在项目管理计划中进行记录，而关于项目管理计划将在第 8 章中进行讨论。

3.4 小 结

一个项目的主要的过程规划问题是设计用于构建软件的开发过程，满足客户的要求。该过程在变更管理过程支持下实现需求变更。

下面列出了从 Infosys 公司采用的方法中吸取的主要过程规划的经验教训：

- 规划项目过程时，从一个标准过程开始。瀑布模型可以作为合理的基础，并将原有的阶段分解成更小的阶段。
- 为了定义一个项目的最佳过程，对标准过程进行裁制以满足项目的约束。首先用主要的项目特征设置裁制的环境。然后执行具体的裁制任务。裁制指南会对此提供一些帮助。
- 有一个独立的需求变更管理过程，对每个变更申请的影响进行评价，并对所有变更的累积影响进行跟踪。关于 CMM，这里描述的过程规划方法满足 CMM-2 的软件项目规划 KPA 的某些要求。裁制方法满足 CMM-3 的集成项目管理 KPA 的某些要求。需求变更管理方法满足 CMM-2 的需求管理 KPA 的某些要求。

3.5 参考文献

[1] B.W. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[2] V.R. Basili and A. Turner. Iterative enhancement, a practical technique for software development. *IEEE Transactions on Software Engineering*, 1(4), 1975.

[3] H. Gomma and D.B.H. Scott. Prototyping as a tool in the specification of user requirements. *Proceedings of the 5th International Conference on Software Engineering*, pp. 333-341, 1981.

[4] B.W. Boehm. A spiral model of software development and enhancement. *IEEE*

Computer, May 1988.

[5] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys*. Addison-Wesley, 2000.

[6] W. Humphrey. *Managing the Software Process*. Addison-Wesley, 1989. ("SEI 软件工程丛书"之《软件过程管理》，瓦茨·汉弗莱著，其英文影印版由清华大学出版社于2002年出版，中译版由清华大学出版社于2003年出版。)

[7] M.P. Ginsberg and L.H. Quinn. *Process Tailoring and the Software Capability Maturity Model*. Technical Report, Software Engineering Institute, CMU/SEI-94-TR-024, 1995.

[8] B.W. Boehm. Improving software productivity. *IEEE Computer*, Sept. 1987.

第 4 章　工作量估计和进度计划

在 2000 年早期，印度报纸和电视热烈地报道新建造的轻型战斗机试飞成功。但是详情报道还是指明了一个严峻的问题：原型产品的提交推迟了 5 年，并且该项目耗去了最初估计成本的 10 倍。全球有很多项目都经历过与此类似的命运，似乎估计的成本和时间永远不足以执行一个项目。在很多工程方面，不正确的估计是项目管理失败的根源；正如软件项目的拙劣记录所明确表明的那样，软件工程也不例外。

在服务业，不正确的估计甚至会造成更大的损失。当 Infosys 公司的总经理南丹·尼尔克尼（Nandan Nilekeni）被问及他希望项目所需具有的一样特性时，他回答道："没有意外"。在客户满意度方面没有意外，在收入和利润方面没有意外。一个项目中后来出现这两方面的意外多半是由于工作量或者进度估计不准确。

对于估计问题既没有特效的办法，也没有现成的办法。然而，项目经理可以采用经以往的经验和数据验证过的指南来提高估计精度。本章讨论了 Infosys 公司的项目经理估计工作量和进度所用的方法，包括实际项目示例和 ACIC 案例研究的估计。

4.1　估计和进度安排概念

在我们讨论 Infosys 公司采用的工作量估计和进度安排方法之前，本节先描述一些与估计和进度安排有关的概念。工作量估计通常在项目的初期进行，即在明确了所要构建的软件时。但是在项目后期有更多的信息可供参考时，也可能要对工作量进行重新估计。

估计一般并不需要非常精确。软件项目中合理的估计很容易变成一种自我实现的预言——人们努力地去满足进度计划（进度计划是从工作量估计中推断出来的）。的确，在软件项目中，甚至没有人能够精确地回答这样一个问题："这种估计是正确的吗？"因为判断某个估计正确性的惟一办法是将它同实际所需的工作量进行比较。由于人类心理学"在有效的时间内尽量扩大工作量"的原则中反映的基本原则，因此我们不能仅仅因为扩展后的工作量满足了估计的工作量就认为

估计是正确的。因此，项目经理的目标是制定合理的估计，既使目标得到满足，又不至于让项目人员感到筋疲力尽。合理估计的误差范围并不很宽，并且取决于人为因素，但它很可能宽得足以为估计留出充分的余地。

工作量估计可以根据直觉，也可以根据以往的经验，但是一种更加科学而理想的方法是采用估计模型。

4.1.1 工作量估计模型

软件估计模型（estimation model）定义它所需的项目特征（需要这些特征的值或者它们的估计）以及用这些值来计算工作量的方法。估计模型不是在真空中工作的，也不能在真空中工作；它需要输入来产生作为输出的工作量估计。在项目初期并且软件本身的细节还不知道时，希望估计模型能够得到需要度量阶段的特征值。

软件的规模是确定要用多少工作量来构建它的主要因素。但是在构思项目的时候，软件的最终规模是不知道的，并且软件也不存在。因此，如果工作量估计模型要用到软件规模，则必须在最初的估计中做出估计。

一种常见的方法是用一个简单的方程从估计的规模求出估计的总工作量。该方程可以根据以往的工作量和规模数据经回归分析而得到。[1,2]然后，在知道项目的总工作量之后，各阶段或者任务的工作量就可以根据它们占总工作量的百分比而得到。

采用这种自顶向下的方法（top-down approach）进行估计的模型已提出了很多[1,3]，其中 COCOMO 模型最为出色。[3,4]使用功能点（而不是 LOC）作为规模单位的模型也建立了一些。[5,6]在这些模型中，基于这些因素对估计进行精化后，还可以融入其他因素。这是 COCOMO 模型采用的方法。[1]另一种方法根据这些参数调整系统的规模，这种方法以功能点为单位计算软件规模。

另一方面，在自底向上的方法（bottom-up approach）中，你首先获得项目各部分估计的规模，然后得到整个项目估计的规模。[1]即，整个项目估计的规模根据各部分估计的规模而求得。有一种自底向上的方法需要使用某种基于任务的估计。在这种方法中，首先列出主要任务，然后估计每种任务的工作量。从这些估计中，就可以求得整个项目的工作量。

自底向上的方法本身求助于工作量的直接估计，在项目被划分成更小的任务

时，就有可能直接估计它们所需的工作量。虽然项目规模在确定一个项目中许多任务的工作量方面很有用，但这种方法的主要优点是它不需要软件的显式规模估计。相反，它要求一个项目任务列表，这在某些情况下是很容易准备的。自底向上估计方法的风险是有可能在任务列表中忽略某些重要的任务。在直接估计任务的工作量时，可以证明直接估计某些经常性管理任务方面是不容易的，诸如项目管理，它们遍及整个项目并且不如编码或者测试那样经过明确定义。

自顶向下估计方法和自底向上估计方法都需要有关项目的信息：规模（自顶向下估计方法）和任务列表（自底向上估计方法）。在很多方面，这些方法是互补的。[1]如果能够得到更多关于项目的信息，或者随着项目的进展，这两种估计都会更加正确。例如，在给定非常概略的需求下，规模估计就比较困难；而在设计完成后却会变得极其容易；而在开发了代码以后，工作量估计不仅更加容易而且也更加精确。因此，估计的精度取决于何时估计工作量，随着关于项目的信息越来越多，估计的精度也不断增加。

4.1.2 进度估计

在工作量已经知道或者确定以后，根据项目投入的资源（人员）量，就可以制定各种进度计划（即项目的持续时间）了。例如，对于一个工作量估计是 56 人月的项目，则一个 7 个人工作 8 个月的进度计划是可能的。一个由 8 个人工作 7 个月的进度计划也是可能的，当然由 6 个人工作 9 个多月的时间也是可以的。

然而，众所周知，在一个软件项目中，人力和月工作时间并非总是可以互换的。[8]例如，在上述例子中，一个 56 个人工作 1 个月的进度计划是不可能的，尽管其工作量“满足”需求。同样，也不会有人制定让 2 个人工作 28 个月的进度计划。换句话说，在工作量确定以后，通过合理分配项目所需的人员，就可以灵活地设置进度计划。但是这种灵活性也不是毫无限制的，事实表明在工作量和进度计划之间不存在符合经验数据的简单方程。

“拉长”进度很容易：你只要投入更少的人员即可（尽管在很长一段时间内完成项目可能没有多大价值）。然而，压缩进度却不容易。前面所举的例子已明确表明：你不能将一个 56 人月的项目进度压缩成 1 个月完成的项目，即使增加足够的人员也不行。一般而言，进度被压缩到“正常”范围以外就会增加工作量；通过投入比所需的最佳资源更多的资源，必将导致资源的浪费、执行更多的重复工作等等。

有一些方法讨论了进度压缩对总工作量的影响。然而，为了评价这种影响，你必须首先定义一个项目的正常进度。

一种确定正常（或者额定）进度的方法是使用合适的函数从工作量中确定它。而一种确定合适的函数的方法是研究已完成项目的模式。例如，你可以画出已完成项目的工作量和进度的分布图，然后通过此分布图拟合一条回归曲线。此曲线一般是非线性的，因为进度并非随工作量线性增长的。然后你就可以用曲线方程确定已知估计工作量的项目的进度计划。许多模型遵循这种方法，勃姆写的书总结了各种各样的模型。

4.2 工作量估计

在 Infosys 公司，工作量估计一般在需求分析之后进行。即，在项目经理估计工作量之际，已经对需求有了很好的理解。例如，需求分析阶段有时可以作为软件开发项目的一个独立的项目进行执行。

在 Infosys 公司，提出了多种估计方法，本节将讨论其中的一些方法。项目经理可以根据工作性质选择使用任何一种估计方法。在某些情况下，项目经理可以用多种方法进行估计，要么为了验证其主要估计方法的正确性，要么为了减少风险：尤其在以往同类项目的数据有限的情况下，更要采取多种方法进行估计。

4.2.1 自底向上的估计方法

因为 Infosys 公司所承担的项目千变万化，所以自底向上方法倍受青睐，并且公司也推荐使用这种自底向上的方法。公司采用一种任务单元(task unit)方法[1]，这种方法存在一定的局限性，但是通过使用以往的数据和过程能力基准（参见第 2 章），克服了其中的某些局限性。

在任务单元方法中，项目经理首先将开发项目分割成主要程序（即单元）。然后根据某种标准将每个程序单元分类为简单程序、中等复杂程序或者复杂程序。对于每个分类单元，项目经理定义编码和自我测试（统称为构建工作量）的标准工作量。标准的构建工作量可以根据一个同类项目的以往数据、内部现有的指南，或者综合各种方法而得到。

在已知上述三种复杂性类别的单元数并选定了每个程序估计的构建工作量

后，就可以求得项目构建阶段的总工作量。根据构建工作量，其他阶段和任务所需的工作量就可以根据它们占编码工作量的百分比而求得。根据过程能力基准或者过程数据库，就可以知道一个项目的工作量分布情况。项目经理用这种分布情况就可以确定其他阶段和任务的工作量。根据这些估计，就可以求得项目的总工作量。

这种方法必须做到经验和数据的有机结合。如果没有合适的数据（例如，如果你从事一种新的项目类型），则可以在对项目进行分析以后，并且在知道各种程序单元的情况下，根据经验估计构建工作量。估计了构建工作量之后，根据以往项目的工作量分布数据，就可以求得其他任务的估计工作量。该方法甚至还能解决那些在项目初期很难列出并且确实很费劲的任务的工作量估计。在项目的工作量分布中，“其他”分类通常用来处理混杂的任务。

综上所述，估计过程可以用如下步骤进行概括：

（1）标识出系统中的程序，并将它们分类为简单程序、中等复杂程序或者复杂程序（Simple, Medium, or Complex, 简称 S/M/C）。尽量采用标准定义或者以往项目的定义进行上述分类。

（2）如果存在项目专用的基准，则根据基准求得 S/M/C 程序的平均构建工作量。

（3）如果不存在项目专用的基准，则用项目类型、技术、语言及其他属性查找过程数据库中类似的项目。然后，用这些类似项目中的数据定义 S/M/C 程序的构建工作量。

（4）如果过程数据库中不存在类似的项目，并且也不存在项目专用的基准，则应用基本过程能力基准中 S/M/C 程序的平均构建工作量。

（5）用项目特有的因素精化 S/M/C 程序的构建工作量。

（6）用 S/M/C 程序的构建工作量及其程序数求得总的构建工作量。

（7）用能力基准中给定的工作量分布情况或者过程数据库中给定的类似项目，估计其他任务的工作量和总工作量。

（8）基于项目特有的因素精化估计。

上述估计过程使用了过程数据库和过程能力基准，有关过程数据库和过程能力基准的讨论参见第 2 章。前文已经提到过，如果某种类型的多个项目正在执行，则可以建立一个项目专用的能力基准。这种基准类似于基本基准，但只用特定项目的数据。这些基准是用于预测该类型的其他项目的工作量的最好的基准。因此，在工作量估计中优先使用它们。

因为有很多因素可以影响一个项目所需的工作量，所以项目特有的因素对工作量估计的影响是很重要的。这里概述的方法不再将参数分成不同的等级，然后确定它们对工作量需求的影响；而是让管理人员确定项目特有的因素对估计的影响。项目经理可以根据他们的经验、团队成员的经验或者在过程数据库中找到的同类项目的数据对此作出调整。

注意，这种将程序分成几类并对每类程序使用平均构建工作量的方法适用于总工作量的估计。然而，在详细的进度安排中——项目经理将每个单元分配给团队成员进行编码，并分配该编码任务的预算时间——需要考虑具体单元的特征以确定比平均时间更多或者更少的时间。

4.2.2 自顶向下的估计方法

与任何一个自顶向下的估计方法相同，Infosys 公司采用的方法从以功能点为单位的软件规模估计开始。功能点可以用标准的功能点计数规则进行计数。另外，如果已知以 LOC 为单位的估计规模，则可以将它转换为功能点规模。

除了规模估计外，自顶向下的方法还要估计生产率。基本方法是以同类项目的生产率水平开始（这种数据可以从过程数据库中得到）或者以标准生产率值开始（这种数据可以从过程能力基准中得到），然后根据需要对那些值进行调整，满足项目的需要。然后，使用生产率估计来计算总的工作量估计。利用总的工作量估计，根据各阶段工作量的百分比分布求得各阶段估计的工作量。（同自底向上的方法一样，这些工作量分布数据从过程数据库或者能力基准中得到。）

总之，自顶向下估计方法涉及如下步骤：

（1）求得以功能点为单位的软件的总规模。

（2）使用项目专用的能力基准、基本的过程能力基准或者同类项目的生产率数据，确定项目的生产率水平。

（3）根据生产率和规模估计求得总的工作量估计。

（4）使用过程能力基准或者同类项目中的工作量分布数据估计各阶段的工作量。

（5）考虑项目特有的因素，精化工作量估计。

同自底向上的估计一样，自顶向下的估计方法允许用项目特有的因素进行精化。实际上，这种精化没有定义确定的因素，而是承认每个项目都有特殊性，它

们可以有一些其他项目所没有的特征。列出这些特征对生产率的影响，或者形式化地表示这些特征对生产率的影响，这也许是不可能的。因此，必须由项目经理确定应当考虑哪些项目因素，以及它们如何影响项目。

4.2.3　用例点方法

Infosys 公司采用的用例点方法基于 Rational 公司的方法，并且类似于功能点方法。如果将用例用于需求规范，则可以应用这种方法。这种方法的基本步骤如下：

（1）将每个用例分类可以分为简单用例、中等复杂用例和复杂用例。这种分类的基础是用例中的事务数，包括辅助脚本。事务定义为任务的原子集，要么全部执行，要么一个都不执行。一个简单的用例有 3 个或者 3 个以下事务，一个中等复杂用例有4～7 个用例，而一个复杂用例有 7 个以上的事务。其中简单用例分配因子为 5，中等复杂用例分配因子为 10，而复杂用例分配因子为 15。表 4.1 给出了这种分类及其因子。

表 4.1　用例复杂性及其因子

用例类型	说明	因子
简单用例	3 个或 3 个以下事务	5
中等复杂用例	4～7个事务	10
复杂用例	>7个事务	15

（2）根据应用中用例因子的加权和求得总的未经调整的用例点（unadjusted use case point，简称 UUCP）。即，对于三种复杂性用例分类中的每一类，首先求出某个特定复杂度的用例数与该复杂度的因子的乘积，然后求出三个乘积的总和即为应用的 UUCP。

（3）调整原始 UUCP 以反映项目的复杂性和项目工作人员的经验。为此，首先通过查阅表 4.2 中给定的因素并用 0～5 评定每个因素，计算技术复杂性因子（technical complexity factor，简称 TCF）。评定值为 0 表明因素与项目无关；而 5 则表示必不可少的因素。对于每个因素，用表中的权重乘以其评定值；然后将这些结果值加起来就得到了 TFactor，然后就可以用如下方程求 TCF：

$$TCF = 0.6 + (0.01 \times TFactor)$$

表 4.2 技术因素及其权重

序号	因素	权重
1	分布式系统	2
2	响应或者吞吐量绩效目标	1
3	终端用户效率（联机）	1
4	复杂的内部处理	1
5	代码必须是可重用的	1
6	易于安装	0.5
7	易于使用	0.5
8	可移植	2
9	易于变更	1
10	并发	1
11	包括特殊的安全特征	1
12	提供对第三方的直接访问	1
13	特殊的用户培训设施	1

（4）同样，通过查阅表 4.3 并用 0～5 评定每个因素，计算出环境因子（environment factor，简称 EF）。对于与经验有关的因素，0 表示没有该主题的经验，5 表示专家水平，而 3 则表示平均水平。对于积极性，0 表示没有项目积极性，5 表示积极性高，而 3 则表示平均水平。对于需求的稳定性，0 表示非常不稳定的需求，5 表示不变的需求，而 3 则表示中等稳定程度。对于兼职型工作人员，0 表示兼职型技术人员，5 表示全职工作人员，而 3 则表示平均值。对于编程语言的难度，0 表示易掌握的编程语言，5 表示非常难的编程语言，而 3 则表示一般难度的编程语言。上述因素的加权和给出 EFactor，而 EF 可以通过如下方程求得：

$$EF = 1.4 + (-0.03 \times EFactor)$$

表 4.3 团队的环境因素及其权重

序号	因素	权重
1	熟悉Internet过程	1.5
2	应用经验	0.5
3	面向对象经验	1
4	先导分析人员能力	0.5
5	积极性	1
6	稳定的需求	2
7	兼职工作人员	-1
8	难的编程语言	-1

（5）用上述两个因子，用下式求出最终的用例点（use case point，简称 UCP）：

$$UCP = UUCP \times TCF \times EF$$

进行工作量估计时，一般在整个生命期中为每个 UCP 分配 20 人时，由此得出的估计是很粗的。需要对它按如下步骤做进一步精化。计算有多少个因素的值小于 3，有多少个因素的值大于 3。如果值小于 3 的因素总数很少，则每个 UCP 分配 20 人时是合适的。如果有很多，则每个 UCP 分配 28 人时。换句话说，每个 UCP 分配 20～28 人时，而项目经理可以根据各种因素决定使用哪个值。

4.2.4　估计方法的有效性

分析估计方法有效性的一般方法是比较估计的工作量与实际工作量。前文已经讨论过，这种比较只能给出一般的估计精度；它不能指明估计方法有多优。为了获得这种信息，必须研究估计对程序员的影响（例如，他们是“过分紧张”还是“未充分发挥作用”。）然而，扩展的实际工作量与估计的工作量进行比较确实能够表明估计方法的有效性。

在第 3 章中已经讨论过，对于已完成的项目，过程数据库既包含估计的工作量信息，也包含实际的工作量信息。图 4.1 所示的是一些已开发完成项目的估计工作量和实际工作量的分布图。

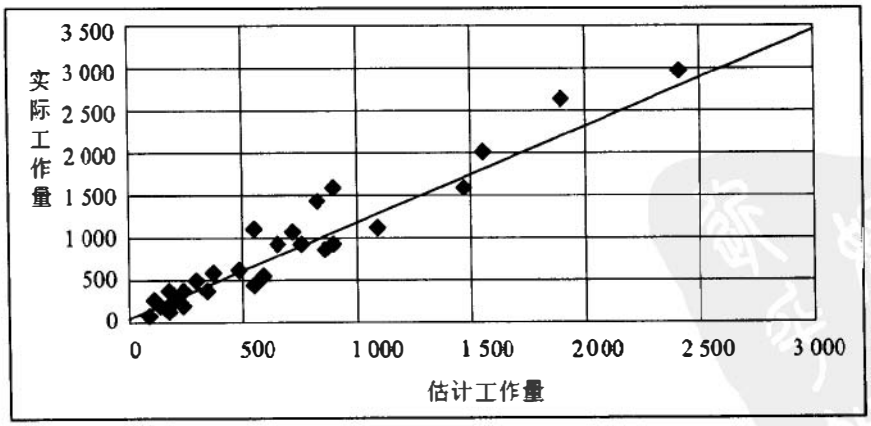


图 4.1　实际工作量与估计工作量

如图所示，估计方法效果很好；大部分数据点接近于图中的45度线（如果所有估计都匹配实际工作量，则所有点都将落在45度线上。）图中的数据还表明：50%以上的项目在估计工作量的25%以内完成。然而，数据表明估计工作量通常低于实际工作量；注意大部分点在45度线以上，而不是在45度线以下。即，人们更易于出现估计不足——这通常使软件业倍受折磨。实际工作量平均比估计的工作量高25%。总之，尽管还有改进的余地，但是估计方法已是相当有效了。

4.2.5 ACIC项目的工作量估计

现在，我们通过例子说明估计方法，说明其在ACIC项目中的应用。在我以前的书中还可以找到另外2个例子[10]。ACIC项目采用用例驱动的方法。因此，主要根据用例进行分解，而不是根据模块。为了对用例进行分类，项目经理采用了分类标准。表4.4列出了26个用例及其复杂性。

表4.4 ACIC项目中的用例

用例编号	说明	复杂性	用例编号	说明	复杂性
1	屏幕导航	复杂	14	更新一个账户的细节	中等复杂
2	更新个人详细资料	中等复杂	15	维护一个账户的任务	复杂
3	增加地址	中等复杂	16	维护一个账户的备注	简单
4	更新地址	复杂	17	查看历史的团体细节	复杂
5	删除地址	复杂	18	查看账户的历史细节	复杂
6	添加电话号码	中等复杂	19	查看选项等级和服务选项的历史	简单
7	更新电话号码	复杂	20	查看任务和备注的历史	简单
8	删除电话号码	复杂	21	查看角色的历史	复杂
9	添加E-mail	中等复杂	22	查看账户细节	简单
10	更新E-mail	中等复杂	23	查看账户的所有权	复杂
11	删除E-mail	中等复杂	24	查看一个账户的紧迫的定单	复杂
12	更新一个团体的就业情况	中等复杂	25	关闭/重新激活账户	简单
13	更新一个团体的财务情况	中等复杂	26	智能更新ACIC的商业伙伴	复杂

为了估计不同类型用例的构建情况，ACIC项目经理借鉴了Synergy项目的数

据，该项目的过程数据库记录项见第 3 章。Synergy 项目有 21 个简单用例、11 个中等复杂用例和 8 个复杂的用例。根据不同用例的详细构建数据，估计平均构建工作量。（总的构建工作量大约为 143 人日。简单用例、中等复杂用例和复杂用例的平均构建时间分别是 1 人日、5 人日和 8 人日。总的估计构建工作量为 140 人日，非常接近于实际的构建工作量。）表 4.5 说明了每种用例类型的平均构建时间和总的构建时间。

表 4.5　ACIC 项目的构建工作量

用例类型	工作量（人日/用例）	单元数	总构建工作量（人日）
简单用例	1	5	5
中等复杂用例	5	9	45
复杂用例	8	12	96
总的构建时间			146

为了估计各阶段的工作量分布情况，项目经理借鉴了 Synergy 项目的分布情况。因为以前的项目没有需求分析阶段，所以必须对工作量分布情况加以修改。表 4.6 给出了每个阶段的估计工作量和总工作量。

表 4.6　ACIC 项目的估计工作量

任务	估计的工作量	
	人日	占总工作量的百分比（%）
需求分析	50	10
设计	60	12
构建	146	29
集成测试	35	7
回归测试	10	2
验收测试	30	6
项目管理	75	15
配置管理	16	3
培训	50	10
其他	40	6
估计的工作量	501	100

在这个项目中，除了按自底向上估计方法进行估计外，项目经理还采用了用例点方法。前文已经讨论过，每个简单用例分配因子为 5，每个中等复杂用例分配因子为 10，每个复杂用例分配因子为 15，所以据此可以得出 UUCP。因为简单

用例、中等复杂用例和复杂用例分别有5个、9个和12个，所以可以得到：

$$UUCP = 5\times5 - 9\times10 + 12\times15 = 295$$

因为需要考虑各种因素，首先，ACIC 项目经理要为技术复杂性分配权重，并求得技术复杂性因子。他选择了如下因子值（按表4.2给定的顺序）：4，3，5，3，4，5，5，0，4，1，2，0和5，计算TFactor值为40（8+3+5+3+4+2.5+2.5+0+4+1+2+0+5），TCF为1.0。接着，他计算环境因子。他为环境因素分配了如下因子值：3，1，3，4，5，5，0和3；最终的EFactor为22（4.5+0.5+3+2+5+10+0-3），EF为0.74。根据上述条件，他计算出总用例点如下：

$$UCP = 295\times1.0\times0.74 = 218.3$$

使用标准的工作量值20人时/UCP，他求得了工作量估计如下：

218×20=4 360人时=499人日（8.75小时/天）

或者

513人日（8.5小时/天）

这些估计惊人地接近于当初的估计，增强了项目经理对估计的信心。（结果表明，这个项目的估计的确非常精确，你可以在第12章的结束报告中看到。此外，在所有里程碑中，与计划的工作量相比，超出的工作量是很小的，这你可以在第12章中给定的里程碑分析中看到。）

前文已经提到过，该项目使用了RUP的迭代过程。因为很多迭代过程都会牵涉到设计、分析和构建阶段，所以按阶段的工作量估计本身不能作为规划的直接输入条件。规划工作量时，项目经理必须估计每个迭代过程的工作量。为了求得此工作量，他以前面求得的所有估计开始。根据各迭代过程选定的用例和RUP方法给定的指南，设计、构建和测试工作量被分解成细化和构造两部分。项目管理、CM和其他成本保持不变。表4.7表明了迭代过程的工作量分配。

4.3 进度计划

Infosys公司的进度安排任务可以分解成两个子任务：一个子任务是确定总的进度计划（项目持续时间）及主要里程碑，另一个子任务是制定各种任务的详细进度计划。

表 4.7　ACIC 项目按迭代过程的工作量分配情况

迭代过程	估计的工作量	
	人日	占总工作量的百分比（%）
项目开始	25	5
初始阶段	24	5
细化阶段：第 1 次迭代	45	9
细化阶段：第 2 次迭代	34	7
构造阶段：第 1 次迭代	27	5
构造阶段：第 2 次迭代	24	5
构造阶段：第 3 次迭代	21	4
转换阶段	110	22
项目收尾	10	2
项目管理	75	15
配置管理	16	3
培训	50	10
其他	40	8
总的估计工作量	501	100

4.3.1　总的进度计划

在本章前面已经讨论过，通过控制人员分配情况可以灵活地确定进度计划，但是这种灵活性是有限的。由于这种灵活性的存在，因此建立严格的进度计划指南可能并非可取；严格的指南失去了项目或者客户应有的灵活性优势。此外，项目进度计划通常在更大的商业计划环境中进行确定，这就会强加一些进度要求。只要有可能，你应当灵活地利用进度来满足这种需求。一种可取的方法是使用进度计划指南来检查进度计划的灵活性，而不是用来确定进度本身。

图 4.2 给出了 Infosys 公司的一些已完成的开发项目的进度和工作量的分布图，同时给出了此分布图非线性回归拟合曲线。

图 4.3 中的曲线方程如下：

$$进度=23.46(工作量)^{0.313}$$

从数据点的分布情况可以看出，显然进度不仅仅是工作量的函数。然而，所确定的进度可以用作进度合理性的标准或者凭据，这也可以根据其他因素来决定。同样，同类项目的进度和工作量数据可以用来检查任何建议进度的合理性。

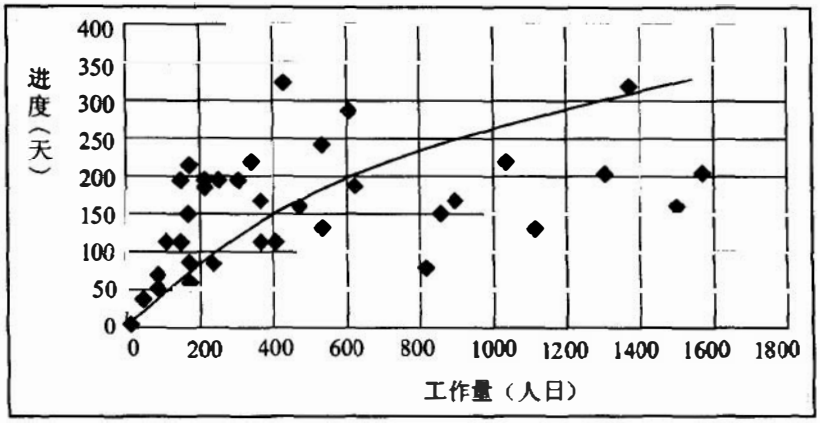


图 4.2 进度与工作量的函数

项目经理通常用所谓的平方根检验经验方法来检查中等规模项目的进度。其原则是建议的进度应当在以人月为单位的总工作量的平方根左右；如果$\sqrt{\text{工作量}}$资源分配给项目，则进度能够得到满足。例如，如果估计的工作量为 50 人月，则一个由 7～8 个专职人员工作 7～8 个月的进度计划比较合适。

由于进度和资源之间的关系，只有在项目所属的业务部门的主管同意提供所需的资源时，才能接受一个进度计划。如果没有所需的资源，则必须对进度计划作出调整。在接受一个进度计划之前还要检查项目的依赖性。如果项目执行依赖于外部因素（诸如另一个项目的完成或者存在某种软件），则必须对进度作出调整以适应这些因素。

一旦确定了项目的总期限，就必须确定主要里程碑的进度计划。为了确定里程碑，首先必须理解一个项目中经常出现的人力分配情况。软件项目中的人数基本上遵循瑞利（Rayleigh）曲线。[9,11]在项目开始和结束时，项目的工作人员很少；而在项目中期前后达到团队最多的人数（peak team size，简称 PTS）。这种行为之所以发生是因为在需求分析和设计的初始阶段只要几个人参加就可以了。而在编码和单元测试时人力资源达到顶峰。此外，在系统测试和集成时，所需的人员更少。在许多情况下，人员分配情况不会经常变动，但是近似于瑞利曲线：在开始时分配几个人员，在构建阶段达到顶峰，然后只需留下少数人进行集成和系统测试。如果将设计、构建和测试作为三个主要阶段，并且它们的需求分析已经完成，则项目的人力分配情况通常类似于图 4.3所示的函数。

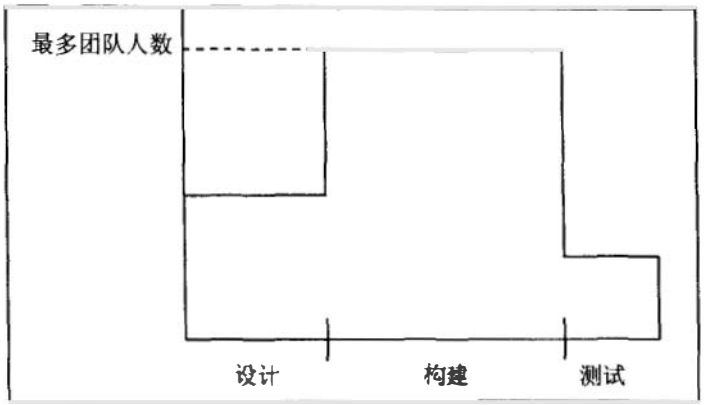


图 4.3　典型项目的人力分配情况

在 Infosys 公司，通常遵循这种资源分配方法。在开始和结束阶段只分配很少一部分人，而在构建阶段分配最多人数。在构建阶段，通常达到项目的 PTS。

为了使进度安排更加容易，尤其是更小规模的项目，所有所需的人员通常都在项目开始时进行分配。这种方法会导致一些人在开始和临近结束时无所事事。这一空闲时间通常用于培训。通常需要对项目所用的技术和业务领域进行培训，并且这一培训会占用大量工作量，这可以从 PCB 给定的工作量分配情况中看到。同样，项目临近结束时的空闲时间可用于文档处理和其他结束任务。

进度分配不同于工作量分配。对这三个主要阶段，构建阶段占进度的百分比少于其占工作量的百分比，因为这一阶段牵涉到更多的人员。同样，设计和测试阶段占进度的百分比大于它们占工作量的百分比。确切的进度依赖于计划的人力分配。在给定一个阶段的工作量估计的情况下，知道了人力分配后就可以确定各阶段的持续时间。

一般而言，设计大概占进度的 40%（20%用于概要设计，20%用于详细设计），构建占 40%，而集成和系统测试占 20%。设计、构建、集成和测试阶段的人力分配分别是 1∶2∶1（因此这些阶段的工作量分配为 1∶2∶1）。这些指南提供了确定里程碑的标准，当然里程碑也可以根据其他约束进行设置。

必须承认，即使将一个人专门分配给一个项目，他通常也会从事一些既耗时又于项目无益的任务。这些任务包括休假、公司任务、基本培训（不是项目特设的培训）、其他项目的评审等等。

4.3.2 估计方法的有效性

与工作量估计一样，一种检查进度估计好坏的方法是画出估计的进度与实际的进度，并查看这些点离 45 度线有多近。如果所有点都非常接近 45 度线，则可以认为进度安排方法是有效的。图 4.4 给出了以前完成的开发项目的这种图。

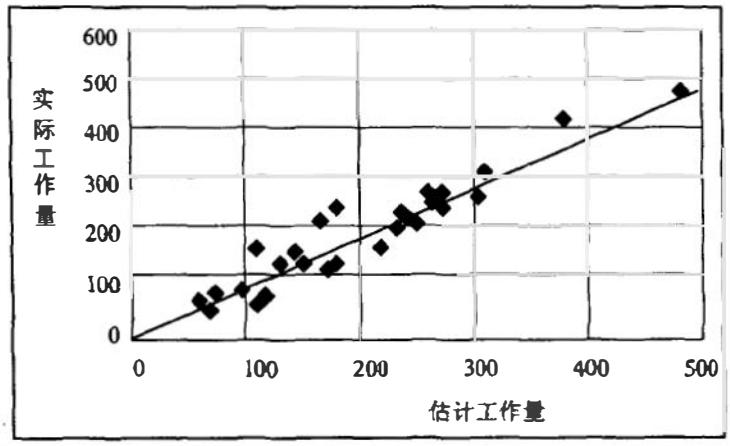


图 4.4 估计与实际进度的比较

可以看出，进度安排方法将产生与实际进度非常匹配的进度。但是记住，其他因素（4.2节所讨论的因素）可以确定是否满足了估计的进度。

4.3.3 详细的进度计划

一旦确定了里程碑和资源，就该设置详细的进度安排。项目经理将任务分解成可调度的任务，形成一个层次结构。对于每个详细的任务，项目经理估计完成它所需的时间并分配合适的资源，以满足总进度。在分配资源方面，必须考虑各种因素，诸如团队成员的休假计划、他们个人的成长计划和职位升迁、他们的技能和经验、培训和顾问需要、任务的紧急程度，以及从某个任务中获取经验以后对项目有什么价值。

进行每一级的精化时，项目经理在详细的进度计划中根据工作量估计检查总任务的工作量。如果需要，对详细估计进行调整。例如，可能将详细设计阶段分解成多个任务，诸如开发每个模块的详细设计、评审每个详细的设计、修复所发

现的故障，等等，并且还可以对上述任务做进一步分解。然后，为这些任务安排时间，并分配在某个时间段内所需的资源。

如果这种详细的进度计划与详细设计的总进度和工作量估计不一致，则必须变更详细进度计划。如果发现最好的详细进度计划不能满足里程碑的工作量和进度，则必须修改早期的估计。因此，进度计划是一个迭代过程。

通常，项目经理将任务细化到一定程度，使最低级的任务不会对一个单独的资源占用很多天。此外，还要增加基本任务，诸如项目管理、协调、数据库管理和配置管理等。这些任务对进度基本上没有直接影响，因为它们是正在进行的任务，而不是可调度的任务。另外，由于它们消耗资源，因此必须在项目的进度计划中考虑它们。

项目经理几乎不能一次性完成整个项目的详细的进度计划。一旦确定了总进度，就有可能只在每个阶段开始时执行那个阶段的详细进度计划。

对于编制详细的进度计划，项目经理通常采用 Microsoft Project（MSP）或者电子表格。对于每个最低级的任务，它们规定工作量、持续时间、开始日期、结束日期和资源。对于每个任务，他们还规定任务代码（将在第 7 章中进行深入讨论）、程序代码以及模块代码。他们还可以规定任务间的依赖性，要么是内在的依赖（例如，你只能在程序编码完成以后才能对它执行单元测试计划），要么是与资源有关的依赖（同一个资源分配给两个任务）。

一个详细的项目进度计划绝不可能是静态的，有时可能需要做某些变更，因为项目的实际进展可能不同于计划，可能是为了满足变更申请的需要增加新的任务，或者因为发生其他不可预见的情况。总之，只在需要的时候才变更。

根据 MSP 或者其他一些工具所记录的那样，最后的进度计划是最“有效的”项目计划文档。在项目执行期间，如果必须修改计划，必须完成其他任务，则在作出决策以后，任何修改都反映在详细的进度计划中。因此，详细的进度计划成为跟踪任务和进度的主要文档。详细进度计划也是项目监督的主要输入，我们将在第 11 章进行讨论。

4.3.4　ACIC 项目的进度计划

让我们以 ACIC 项目为例。（有关其他案例研究的进度计划，参见我以前所写的书。[10]）前文已经讨论过，ACIC 项目估计的工作量是 501 人日，即大约 24

人月。客户要求用大约 5.5 个月完成该项目（从 5 月 15 日到 11 月 3 日）。因为这一期限大于以人月为单位的工作量的平方根，并且因为需求收集必须在项目开始前完成，所以接受了这一进度计划。（我们对这个进度计划的资源需求也进行了估计，并于第 8 章的项目管理计划中给出。）

根据各阶段的工作量估计及这个阶段可以完全占用的资源量估计确定里程碑。在 ACIC 项目中，项目经理列出每个阶段的主要任务，并为它们分配资源。根据资源分配，就可以确定该阶段的总进度计划和工作量。如果该阶段的总工作量不满足工作量估计，则必须对资源分配不断地进行调整，直到总工作量匹配工作量估计为止。然后，根据任务分配求得的总进度，就作为该阶段的进度计划。（第 8 章给定的项目管理计划规定了里程碑。）表 4.8 表示 ACIC 项目的高层（high-level）进度计划。该进度计划从项目最终的详细进度计划中自动推出。

表 4.8 ACIC 项目的高层进度计划

任务标识符	任务	持续时间(天)	工作(人日)	开始日期	结束日期
1	ACIC 开发进度计划	139.56	559.93	4/3/00	11/3/00
2	项目开始任务	33.78	24.2	5/4/00	6/23/00
29	正常任务	87.11	35.13	6/5/00	10/16/00
74	培训	95.11	49.37	5/8/00	10/16/00
99	组织任务	76.89	12.9	5/22/00	9/15/00
104	知识分享原则	78.22	19.56	6/2/00	9/30/00
110	初始阶段任务	26.67	22.67	4/3/00	5/12/00
114	细化阶段：第 1 次迭代	27.56	55.16	5/15/00	6/23/00
157	细化阶段：第 2 次迭代	8.89	35.88	6/26/00	7/7/00
198	构造阶段：第 1 次迭代	8.89	24.63	7/10/00	7/21/00
228	构造阶段：第 2 次迭代	6.22	28.22	7/20/00	7/28/00
256	构造阶段：第 3 次迭代	6.22	27.03	7/31/00	8/8/00
290	转换阶段任务	56	179.62	8/9/00	11/3/00
323	2.0 版本窗口调整的代码发布	26.67	39.11	8/14/00	9/22/00
331	后端主机 3.0版的工作	4.44	6.44	8/14/00	8/18/00

在该表中，任务标识符（ID）是 MSP分配的序号。任务 ID 表明最后的进度计划中的总任务数多于 330 个，并且这些高层任务中的每一个都包含许多可调度的任务。第 1 个任务是总项目，持续时间约 140 天，工作量为 560 人日。（该进度计划源自项目的最后进度计划，它结合了要完成的变更；该表中的最后两个任务

是两个主要变更。）

这种高层进度计划不适于资源分配和详细规划。进行详细进度计划时，这些任务被分解成可调度的任务。这样，进度计划就变成了项目的一个任务检查表。前文已经提到过，这种自顶向下任务的“爆发”不是全部在开始时完成的，而是在项目期间多次完成的。

表 4.9 说明了 ACIC 项目的构造阶段：第 1 次迭代的详细进度计划。对于每个任务，该表规定了模块、程序和任务的代码，以及持续时间、工作量等等。模块和程序列代表已经在工作的模块和程序。任务代码表示正在执行的任务。（标准的组织范围的代码在第 7 章中进行讨论。）

表 4.9　ACIC 项目的部分详细进度计划

模块	程序	活动代码	任务	持续时间（天）	工作量（天）	开始日期	结束日期	完成量（%）	资源缩略语
		PRS	需求	8.89 天	1.33 天				
		PDDRV	设计评审	1 天	0.9 天				
		PDDRW	设计评审后的修改	1 天	0.8 天				
	UC17	PCD	查看团队细节的历史信息，UC17	2.67 天	1.87 天				
历史信息	UC17	PCDRV	代码走查，UC17	0.89 天	0.27 天				
历史信息	UC19	PCDRV	代码走查，UC19	0.89 天	0.27 天				
		PCDRW	代码走查后的修改	0.89 天	2.49 天				
		PUTRW	测试后修改	0.89 天	0.71 天				
历史信息	UC17	PUT	测试，UC17	0.89 天	0.62 天				
历史信息	UC19	PUT	测试，UC19	0.89 天	0.62 天				
配置		PCM	调解	0.89 天	2.49 天				
管理		PPMPT	进度安排和跟踪	7.11 天	2.13 天				
质量		PPMPT	里程碑分析	0.89 天	0.62 天				

有时，还要规定任务的前置任务（任务所依赖的任务），但这里不做介绍。此信息有助于确定关键路径和关键资源。

对每个任务，完成程度在“完成量（%）”列给出。此信息用于任务跟踪，这在第 11 章中做进一步讨论。详细的进度计划还规定那些分配了任务的资源。

表 4.9 忽略了任务号。表 4.8 指出，ACIC 项目最终的进度计划有 330 多项，最低层的任务是可调度的任务。

4.4 小　结

工作量估计的基本目标是产生在大多数时间内适用的合理的估计。下面是从Infosys公司采用的估计和进度安排方法中吸取的主要经验教训：

- 使用过去的数据进行估计。宁可用同类项目的数据而不用基本的过程能力数据。估计时采用一个模型，但是允许灵活地调整估计以满足项目特有的因素。
- 在不同的情况下使用不同的模型。在知道项目细节的情况下，采用自底向上的估计方法比较有效。如果你可以估计规模和生产率，则采用自顶向下的估计方法；而在使用基于用例的开发方法时采用用例点方法。
- 对于总的进度计划和高层里程碑，使用现有的灵活性满足建议的日期。一旦确定了总的进度计划和里程碑，就可以根据各阶段的工作量估计求得每个阶段的资源需求。
- 详细的进度计划是一项动态任务；分配任务时要考虑人的问题。没有必要在开始时精炼进度计划。你可以在整个进度计划中根据需要制定任务的细节。
- 详细的进度计划形成了项目中计划的任务表。该文档记录了项目中计划的所有任务，随后可以用它跟踪任务。

根据CMM模型的观点，正确的工作量和进度估计方法是CMM-2的软件项目规划KPA的一项需求。在CMM-4中，希望用以往数据进行评估的做法有所增加；并且为了满足量化过程管理的KPA，必须拥有一种良好的估计方法。CMM-3的集成软件管理KPA也要求有良好的估计方法对项目进行规划。本章所讨论的方法符合与这些KPA的估计有关的需求。

4.5 参考文献

[1] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[2] S.D. Conte, H.E. Sunsmore, and V.Y. Shen. *Software Engineering Metrics and Models*. Benjamin/Cummings, 1986.

[3] V.R. Basili, *Tutorial on Models and Metrics for Software Management and Engineering*. IEEE Press, 1980.

[4] B. Boehm. Software engineering economics. *IEEE Transactions on Software*

Engineering, 10(1), 1984.

[5] C.F. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5), 1987.

[6] J.E. Matson, B.E. Barrett, and J.M. Mellicham. Software development cost estimation using function points. *IEEE Transactions on Software Engineering*, 20(4), 1994.

[7] A.J. Albrecht and J.R. Gaffney. Software function, source lines of code, and development effort predication: A software science validation. *IEEE Transactions on Software Engineering*, 9(6), 1983.

[8] F. Brooks, Jr. *The Mythical Man Month, Anniversary Edition.* Addison-Wesley, 1995.

[9] L.H. Putnam and W. Myers. *Industrial Strength Software: Effective Management Using Measurement.* IEEE Computer Society Press, 1997.

[10] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys.* Addison-Wesley, 2000.

[11] L.H. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, 4(4), 1978.

第5章 质量规划

直到最近几年，软件工程才经受到了制造业在很早以前就已经受过的悲剧般的质量概念——质量是在装配/开发过程结束之后但是在产品即将交付使用之前完成的任务。一些有质量意识的项目经理通常计划在开发过程结束后进行系统测试，但是在开发时却不重视质量控制任务，而另一些项目经理甚至连正确的系统测试计划都没有，这是很常见的情况。结果怎样呢？系统测试往往会发现比期望的故障多得多的故障。而修复这些故障所需的工作量也往往比计划的要更多，最终导致软件的bug很多，并且交付日期也比计划的要晚。

随着情况的好转，项目经理开始作评审与单元测试的规划。但他们不知道如何判断这些措施的有效性及其所需的成本。换句话说，项目仍然缺乏明确的质量目标、令人信服的目标实现计划，以及监督诸如单元测试等质量控制任务的有效性的机制。

在合理利用度量标准和以往数据的情况下，就有可能在对待质量问题上如同对待另外两个重要的参数那样：工作量和进度。即，你可以设置定量化的质量目标，以及有助于跟踪项目的质量目标实现情况的子目标。

本章讨论Infosys公司的项目经理如何设置项目的质量目标，如何制定实现这些目标的计划（这些计划用中间质量目标监督质量目标的实现情况）。在描述Infosys公司采用的方法之前，我们先简单地讨论一些有关质量管理的基本概念。

5.1 质量概念

确保最终交付的软件高质量是项目经理主要关心的事情之一。但是，如何定义软件质量呢？软件质量概念不容易定义，因为软件有很多质量特征。[1]实际上，质量管理通常围绕故障而展开。因此，我们用已交付软件的故障密度（delivered defect density）作为软件质量的定义——即，已交付软件中每个单位规模的故障数。此定义是当前业界的事实标准。[2]之所以用它定义软件质量，是为了表明软件项目的目标是使交付的软件具有尽量少的故障。

那么，何谓故障呢？同样，也没有关于故障的通用的且广为接受的精确定义（如果一个软件拼错了一个词，是否认为该软件有一个故障呢？）。通常，我们认为软件的故障是某种使软件表现出与客户的要求或者需要不一致的方式进行运转的问题。

在考虑质量的管理技术之前，必须先知道故障引入和排除周期（defect injection and removal cycle）。软件开发是一种高度依赖于人的任务，因此很容易出现错误。故障可以在软件过程的任何阶段引入。这些引入阶段主要有需求规格说明、概要设计、详细设计和编码。

对于高质量的软件，要求最终产品含有的故障数应尽可能地少。因此，为了发行高质量的软件，必须积极排除故障；这可以通过评审和测试等质量控制任务而实现。因为故障排除的成本随着故障潜伏时间（从一个故障引入到检测到这个故障的时间间隔）的增长而增加[3]，任何成熟的过程都将在每个可能引入故障的阶段之后执行质量控制任务。故障排除任务包括需求评审、设计评审、代码评审、单元测试、集成测试、系统测试和验收测试（我们没有包括计划文档的评审，但是这种评审也有助于改进软件的质量）。图 5.1 表示了故障引入与排除过程。

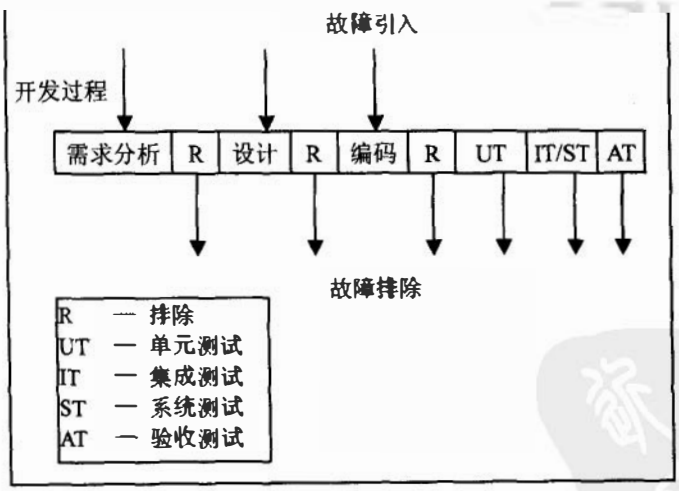


图 5.1 故障引入和排除

质量管理的任务是规划合理的质量控制任务，然后正确地执行和控制它们，以实现项目的质量目标。

5.1.1 质量管理的过程化方法

前文已经提到过，通过执行评审或者测试可以检测故障。但是评审是结构化的、面向人的过程，而测试是以一种试图识别出故障的方式执行软件（或者软件的一部分）的过程。在质量管理的过程化方法中，评审与测试任务的过程和指南是既定的。在一个项目中，这些任务都经过规划（即，执行哪个任务及什么时候执行是确定的）；执行时，根据已定义的过程实现这些任务。总之，过程化方法在已定义的点上执行某些检测故障的过程。

过程化方法不支持声明已排除了多少故障或者故障排除过程结束后的软件质量。换句话说，只执行一系列的故障排除过程不能提供判断它们的有效性或者评价最终代码质量的基础。此外，这种方法高度依赖于过程的质量及过程执行的质量。例如，如果周详地完成了测试计划，并且对该计划进行了全面的评审，则在执行测试后的软件质量与经过测试但是测试计划考虑不周并且评审草草完成的软件质量相比，显然前者要比后者的质量更高。过程化方法的一个主要缺点是项目经理缺乏定量化的手段来评价软件的质量。项目经理惟一能够看到的因素就是质量控制任务是否已经执行了。

5.1.2 质量管理的定量化方法

为了更好地评估故障检测过程的有效性，不仅要问“已经执行这种方法了吗？”还要根据指标数据进行评估。基于这种数据分析，你可以决定是否需要进一步的测试和评审。如果执行项目时应用的控制基于定量数据实现定量化的质量目标，则我们认为项目使用了定量化的质量管理方法。

定量化质量管理涉及到 2 个主要问题：一个是设置定量化的质量目标，另一个是定量化地管理软件开发过程以满足这一质量目标（具有高可信度）。

一个良好的质量管理方法应当在项目早期发出警告信号，而不仅仅是在临近项目收尾时发出警告信号，这是在可选方案有限的情况下采取的策略。为了实现这一目标，关键是预测一些参数在各阶段的值，以便在执行项目时通过对它们的控制就可以确保最终产品具有期望的质量。如果可以实现这种预测，则可以用收集到的实际数据判断过程是否得到了有效的应用。采用这种方法，故障检测过程在过程执行完后并没有结束；相反，有关过程执行的数据用来保证过程执行时完全发挥了其潜能。

一种定量化控制软件质量的方法是使用软件可靠性(software reliability)模型。大多数这样的模型用最后测试阶段的故障数据来估计软件的可靠性。这些模型可以表明可靠性是否可接受，是否需要进一步的测试。可惜的是，它们没有提供项目早期的中间质量目标，并且它们还存在其他局限性。总之，虽然这种模型在估计软件产品的可靠性方面是有用的，但是它们对质量管理的作用却是有限的。(有关的更多信息，请参见可靠性模型。[4,5,6])

关于软件的另一种公认的质量概念是故障排除效率（defect removal efficiency，简称DRE)。对于一种质量控制（quality control，简称QC）任务，我们将故障排除效率（DRE）定义为：QC 任务检测到的故障数占现有总故障数的百分比。[5]项目的整个生命期的DRE（即，在交付软件前执行的所有任务）表示过程的过程中效率（in-process efficiency of the process)。如果已知项目的总故障引入率，则整个生命期的DRE也定义了软件的质量（已交付软件的故障密度)。

虽然故障排除效率是评估一个过程和鉴别出改进领域的一个有用的指标，但它本身不适于质量管理，主要原因是一个QC任务或者整个过程的DRE只有在项目收尾时（即，在知道了所有的故障及其起源时）才可以计算。因此，它没有提供项目执行时直接控制质量的方法。

另一种定量化质量管理的方法是故障预测（defect prediction)。这种方法根据已交付软件的故障密度设定质量目标。通过估计各故障检测任务可能识别出的故障数设定中间质量目标；然后将实际故障数与估计的故障数相比较。

这种质量管理方法非常雷同于工作量和进度的管理（项目成功的另外2个重要的参数)。首先设定软件交付时的质量目标。根据这个目标，估计项目各阶段所选参数的值；即，确立里程碑。这些里程碑的选择要使得在估计都得到满足的情况下，最终软件的质量可能满足期望的目标。在项目执行期间，度量参数的实际值，并与估计的值进行比较，以确定项目是否遵循期望的路线，或者是否需要采取某些措施以确保最终的软件具有期望的质量。

这种方法的有效性取决于你在项目各阶段预测故障数的精确程度。众所周知，故障率遵循与工作量比率相同的模式，两者均遵循瑞利曲线。[5,7,8]换句话说，项目开始时发现的故障数较少，但是随着项目的进展故障数一直在增多，直到单元测试时达到顶峰，然后开始下降。因为过程已经定义了故障检测点，所以你也可以根据各阶段检测到的故障数占总故障数的百分比规定该曲线。根据故障引入率和规模的估计，就可以估计总故障数。这种故障数预测方法既类似于基本故障模型，又类似于IBM的联邦系统划分（Federal Systems Division）[5]方法。

还有一种方法是用统计过程控制（statistical process control，简称 SPC）管理质量（第 7 章简单地讨论了 SPC）。这种方法根据控制界限设置各QC过程期望的绩效，诸如测试和评审等。如果 QC 任务的实际绩效不在界限内，则分析情况并采取合理的措施。控制界限犹如根据以往的绩效预测故障数，但也可以用于在更细粒度上监督质量任务，诸如一个模块的评审或者单元测试。

当采用绩效预测方法并且实际故障数低于目标时，这种方法有太多的不确定性，使你不能肯定地说故障排除过程没有得到正确的执行。因此，你必需查看其他指标以确定原因。[5]换句话说，如果实际数据超出了范围，项目经理将查看其他指标以确定实际情况怎样以及需要采取什么措施。

5.2 定量化质量管理规划

现在让我们讨论 Infosys 公司的项目经理如何采用故障预测方法定量地管理软件质量。（第 10 章和第 11 章将会讨论到，项目也可以在任务级上运用 SPC。）前文已经讨论过，在你规划定量地管理项目的质量时，你将会面临两个关键问题：第一个问题是设置质量目标，第二个问题是预测中间里程碑处的故障数，用于定量地监督项目朝着目标的进展情况。此外，如果项目目标超过现有过程的能力，则项目经理必需计划合理地增强质量过程。让我们看看 Infosys 公司的项目经理是如何执行这些任务的。

5.2.1 设置质量目标

Infosys 公司的项目经理在规划阶段设置质量目标。项目的质量目标通常是验收测试阶段期望发现的故障数。你可以根据用以往的数据计算所得的值设置质量目标，在这种情况下，意味着你将使用标准过程，因此可以期望得到标准的质量结果。可用于设置质量目标的数据源主要有 2 种：同类项目的以往数据和 PCB 数据。

如果你采用同类项目的数据，则可以估计出当前项目在验收测试时发现的故障数等于同类项目在验收测试时发现的故障数乘以该项目估计的工作量与同类项目的总工作量的比。

如果使用 PCB 数据，则可以使用几种方法计算该值。如果设定质量目标为每

个功能点的故障数，则以功能点（前文已讨论过）估计软件的规模，并且期望的故障数是质量值与估计的规模的乘积。具体步骤如下：

（1）设置质量目标：每个 FP 的故障数。

（2）估计项目期望的生产率值。

（3）估计以 FP 为单位的软件规模，其值等于（期望的生产率×估计的工作量）。

（4）估计 AT 故障数，其值等于（质量目标×估计的规模）。

除了根据每个功能点的故障数设置质量目标外，有时根据过程的故障排除效率设置质量目标会更有用。在这种情况下，根据故障引入率、过程中的排除效率和估计的规模，就可以确定验收测试阶段期望的故障数。具体步骤如下：

（1）根据故障排除效率设置质量目标。

（2）根据故障引入率和估计的规模估计总的故障数，或者根据基于工作量的故障引入率和估计的工作量估计总的故障数。

（3）根据总的故障数和质量目标估计 AT 故障数。

5.2.2 估计其他阶段的故障

一旦设定了项目的质量目标，就必需估计出各质量控制任务的故障数，以便你能够定量地控制质量。估计其他阶段故障数的方法类似于估计验收测试阶段故障数的方法。根据可能引入的总故障数的估计，以及 PCB 中给出的故障的百分比分布情况，就可以预报各测试阶段的故障数。另外，你也可以根据同类项目的以往数据预报各阶段的故障数。

至少，你要估计系统测试和集成测试时发现的故障数。之所以选出系统测试，是因为它是主要的测试任务，并且最终的 QC 任务在软件交给客户之前执行。估计系统测试时的故障数，然后将那个数与实际发现的故障数相比较，这种比较将有助于你确定系统测试是否已经充分及软件是否已准备好发行。

对于评审阶段，不用明确地预测故障数，你可以使用评审基准中给定的标准，在执行完评审后立即评估评审的有效性。这些标准根据 SPC 进行确定，允许你评估每个评审的有效性，而不仅仅是评估一个阶段的有效性。第 10 章更为详细地讨论了定量化评审管理。同样，单元测试也有定量化标准。单元测试的监督将在第 11 章作进一步讨论。

5.2.3 质量过程规划

你可以设定一个比同类项目的质量标准更高（或者更低）的质量目标，也可以设定标准过程实现的质量目标。然后，通过采用为项目所设的质量目标，就可以确定更高的质量目标所期望的故障数。另外，在确定了期望的AT故障数之后，通过选择一个不同的AT故障数作为目标，也可以设定质量目标。

如果质量目标基于同类项目的数据并且比同类项目的目标更高，则希望遵循以前的项目所用的同样的过程实现更高的质量目标是不切实际的。如果遵循相同的过程，合理的期望是实现相同的质量目标。因此，如果需要实现某个更高的质量目标，则必需对过程作相应的改进。同样，如果质量目标设得比PCB中给定的质量目标更高，则期望遵循标准过程将达到更高的质量目标是不切实际的。因此，这将需要一个新策略一—通常要结合培训、原型开发、测试、评审，尤其是故障预防。此策略在项目的质量计划中明确声明。这里我们只讨论测试和评审，这是两个主要的质量控制过程。故障预防规划将在本章后面讨论。

一个项目可以有多个不同的测试标准。通过添加或者删除某些测试步骤（这些步骤在项目管理计划中表现为过程偏差（process deviation）），就可以修改总体测试过程。此外，也可以对测试方法进行增强，例如，对测试计划和测试结果进行小组评审。

通常由项目经理选择要被评审的工作产品。实际上，要被评审的文档集因项目的不同而不同，这可以根据质量目标进行调整。如果设定了更高的质量目标，则通过小组评审更大量的程序和测试计划以及更加严格地对详细设计进行小组评审等等，还是有可能实现更高的质量目标。如果选择了这种方法，我们就说选择了满足质量目标的策略。为了详细说明这种策略所需的成本，你可以在项目计划中规定所有要被评审的文档以及这些评审的性质。

你可以用过程能力基准中的数据来估计建议的过程变更对项目计划的工作量和进度的影响。通常，一旦确定了过程变更，就可以基于以往的经验预测它们的影响。这种策略一般是可接受的，因为变更通常很微小。

5.3 故障预防规划

故障预防（Defect prevention，简称DP）任务旨在提高项目的质量和生产率。一般认为各种QC任务不能检测到系统中存在的某些故障，并且不可避免地会在

最终系统中存在这些故障。因此，开发期间引入的故障数越多，则最终交付的系统中残留的故障数也会越多。

我们可以从另一个角度来说明这个问题。一个过程的总的故障排除效率是软件交付之前各种QC任务排除的故障占总故障数的百分比。对于一个稳定的过程，故障排除效率通常也是稳定的。因此，系统中的总故障数越多，则已交付系统中的故障数也会越多。换句话说，故障引入率越高，质量越差。显然，对于一个给定的过程及其故障排除效率，如果构建软件时引入的故障数越少，则最终交付的软件质量就会提高。这种认识称为质量驱动的故障预防。

DP（故障预防）也有生产率方面的优点。前文已经讨论过，软件构建期间的基本故障周期是开发人员引入故障，然后识别出它们并删除它们。换句话说，这好比是先引入某些东西，然后又删除它们。显然，这一故障引入和删除周期是一种浪费——它没有给软件添加任何价值。在这个周期中，开发人员引入某些不需要的东西，最终竟要投入更大的工作量才能删除它们（他们希望删除所有错误）。因此，首先必需做到不引入故障；于是，识别故障并排除它们的工作量就不需要了。换句话说，如果你引入的故障越少，必需排除的故障也越少；而排除故障所需的工作量也会减少，从而提高了生产率。这种认识就是成本驱动的DP。

如何实现DP呢？DP的前提是在引入的故障背后存在某种原因。如果能够发现这种原因，那么就可以着手消除故障或者使它们的影响最小。换句话说，DP通常需要收集关于过去发现的故障数据，对这些数据进行分析以找出引入故障的根源，然后制定方案来解决这些根源。

与任何其他主要的项目任务一样，必须对DP任务进行规划。然而，实际上是在项目已经存在一定数量的故障数据后，分析故障并找到解决方案。为了实现DP，项目经理可以先应用组织级现有的建议，然后基于项目故障数据的分析建立项目专用的建议。在Infosys公司，项目级的故障预防任务采用如下步骤：

- 在项目内组织一个故障预防团队。
- 举行一个动员会议，标识出现有的方案。
- 规划故障预防
 - ✧ 设置项目的故障预防目标。
 - ✧ 如果需要，检查DP团队已受过DP和因果分析培训。
 - ✧ 定义执行故障预防任务的频度。
- 执行故障预防
 - ✧ 在已定义的点上，整理故障数据。

- ✧ 通过执行帕累多（Pareto）分析标识出最常见的故障类型。
- ✧ 执行因果分析，并划分根本原因的优先等级。
- ✧ 识别出根本原因，制定解决方案。
- ✧ 实施方案。
- ✧ 在项目的里程碑处检查 DP 的状态和好处。

- 总结经验知识
 - ✧ 在指标分析报告和 BOK 中，记录你所取得的经验知识和好处。
 - ✧ 作为过程资源的一部分提交 DP 的所有结果。

在质量规划期间，你惟一的任务就是对质量任务进行规划。当项目正在执行或者已执行完时，“执行故障预防”或者“总结经验知识”下的任务都已完成。这里，我们主要讨论与规划有关的任务。第 11 章在项目监督的讨论中描述了“执行故障预防”下的任务。

大部分与 DP 规划有关的任务都是显而易见的。项目经理组织一个执行 DP 分析的团队(显然，项目中的每一个人都必须执行预防故障时要执行的实际方案)。动员会议提请团队成员注意，标识出组织可能有的解决方案。如果需要，对 DP 团队进行 DP和因果分析培训。

设置 DP 目标是一个关键的规划任务。前文已经提到过，DP 是一种实现比标准的组织过程能够实现的质量和生产率更高的策略。因此，如果一个项目经理设定的质量和生产率目标比组织的目标更高，则他可以作为实现该目标的一部分使用 DP。通常，项目经理根据故障引入率的减少量设定 DP 目标，典型值为 10%到 20%。设定了这一比率之后，就可以估计 DP 对质量和生产率的影响了。

其他关键的规划任务确定何时执行 DP 任务以及如何执行 DP任务。尽管项目经理可以作出这些决定，但是 Infosys 公司的基本指导思想是：在大约 20%的程序经过编码、代码评审和单元测试之后必须执行 DP 任务，而在大约 50%的程序经过编码、代码评审和单元测试之后必须再次执行 DP 任务。即，在 20%模块的故障数据已经存在时，执行第一次 DP 任务。如果顺利的话，第一次 DP任务结束后应当产生减少项目其余任务的故障引入率的措施。然后，在 50%模块的故障数据已经存在时，再次执行 DP任务。在这次DP任务中，项目经理可以确定方案是否已起作用，以及是否需要进一步采取措施。这时，应当已经知道所有不同类型的故障及其原因，以及应采取的措施。因此，进一步的分析不会产生太多的新信息。

5.4 ACIC 项目的质量计划

现在，让我们讨论一下 ACIC 案例研究项目的质量规划。（在我以前所著的书中可以找到其他实例，包括一个不同的案例研究。[9]）为了设定其目标和故障估计，ACIC 项目经理借鉴了 Synergy 项目的数据，这是更早些时候为同一个客户开发的同类项目。Synergy 的故障引入率是 0.036 个故障/人时（通过总故障数除以总工作量而求得——总故障数和总工作量都可以在 Synergy 过程数据库中找到）。ACIC 项目经理希望比 Synergy 项目做得更好，并希望将故障引入率减少 10%，显著地好于组织的标准。根据计划的故障引入率，引入的故障数等于 501×8.75×0.033（工作量（以人日为单位）、每人每天的工作时间（以小时为单位）以及期望的故障引入率的乘积）。即，估计整个生命期引入的总故障数在 145 个左右。

质量被定义为验收测试期间的故障密度，或者验收测试前的总故障排除效率。Synergy 在验收测试期间发现 5%故障，ACIC 项目旨在将它减少到 3%，因此验收测试期间发现的故障数为 145×0.03=5（近似）。

生产率目标与 Synergy 相比略微有所提高。进度目标是按时交货，并且期望的质量成本是 32%，这与 Synergy 和组织能力基准相同。表 5.1 给出了 ACIC 项目的所有这些目标。

为了监督和控制项目，ACIC 项目经理需要各阶段检测到的故障数的估计。然后，他就能够将这些估计与实际发现的故障数作比较，并用它们来监督项目的进展。有了引入的总故障数的估计之后，就可以用故障分布率求得上述每个阶段的故障数估计。

表5.1 ACIC 项目目标

目标	值	目标设置基础	组织范围的标准
总故障数	145	0.033 个故障/人时，这比 Synergy 好 10%，后者为 0.036 个故障/人时	0.052 个故障/人时
质量(验收故障数)	5	估计的总故障数的3%或以下	估计的总故障数的 6%
生产率（FP/人月）	57	生产率比 Synergy 提高 3.4%	50
进度计划	按时交付		10%
质量成本	32%	31.5%	32%

为了求得故障分布率，他可以选择应用能力基准数据或者 Synergy 数据。因为 Synergy 没有需求分析阶段，所以他对其分布率作了调整以适应当前的生命期。

本质上，他将单元测试阶段发现的故障百分比从45%减少到40%，验收测试阶段的故障百分比减少到3%（因为这是质量目标），并将需求分析和设计评审阶段的故障百分比增加到20%。这些百分比与能力基准中给定的分布率是一致的。表5.2给出了各阶段检测到的故障数估计。

表5.2 各阶段检测到的故障数估计

评审/测试阶段	估计检测到的故障数	占估计的总故障数的百分比	估计基础
需求和设计评审	29	20%	同类项目（Synergy）和PCB
代码评审	29	20%	同类项目（Synergy）和PCB
单元测试	57	40%	同类项目（Synergy）和PCB
集成测试和回归测试	25	17%	同类项目（Synergy）和PCB
验收测试	5	3%	同类项目（Synergy）和PCB
估计被检测到的总故障数	143	100%	

因为质量目标高于Synergy和组织范围的标准，并且因为生产率目标也有所提高，所以ACIC项目经理必须设计一种实现这些目标的策略，因为标准过程不能实现它们。基本策略具有如下三重目的：

- 采用故障预防
- 对规范和程序员编写的第一个程序执行小组评审
- 使用RUP方法

即，与Synergy采用的过程相比较，ACIC项目采用的过程有所改变，并将实现更高的目标。

基于其他项目的数据，项目经理期望故障预防能够将故障减少10%到20%。这将减少测试和评审后的返工工作量，在生产率方面大约提高了2%（根据Synergy的返工工作量的百分比）。他希望小组评审程序规范和程序员编写的第一个模块将提高总的故障排除效率，同时还希望生产率有所提高。基于有关文献和项目，他希望RUP的使用能够在质量和生产率方面提供效益。表5.3给出了有关的策略和期望的效益。注意，尽管对每个策略期望的效益分别加以描述，但是很难单独地监督各策略的效果。

因为评审是质量过程的一个关键问题，所以应在质量计划中单独描述它们。质量计划应规定开发生命期中执行评审的时机、评审工作产品的时机以及评审的性质。表5.4说明了ACIC项目的这些评审。

表 5.3 ACIC 项目实现更高目标的策略

策略	期望的效益
用标准的故障预防指南和过程预防故障：使用 Synergy 制定的标准进行编码	故障引入率减少 10%-20%，生产率大约提高了 2%
小组评审前面几个程序规范和逻辑上复杂的用例 小组评审设计文档和第一个程序代码	质量提高，由于总的故障排除效率提高了 在生产率方面也有某些效益，因为故障被较早地检测到
引入 RUP 方法，迭代式地实现项目。在每次迭代后执行里程碑分析和故障预防任务	故障引入率大约减少了 5%，而总体生产率大约提高了 1%

表 5.4 ACIC 项目的评审

评审时机	评审项目	评审类型
项目规划结束时	项目计划 故障控制系统计划 项目进度计划	小组评审 软件质量顾问评审 软件质量顾问评审
项目规划结束时	CM 计划	小组评审
90%需求分析结束时（这必须在第 1 次细化迭代结束时）	业务分析和需求规范文档 用例目录	小组评审
90%设计结束时（这必须在第 2 次细化迭代结束时）	设计文档 对象模型	小组评审
每次迭代开始时	迭代计划	个人评审
详细设计结束时	复杂的程序规范和第一次产生的程序规范，包括测试案例、交互图	小组评审
前面几个程序编码以后	代码	小组评审
自我测试一个过程以后	代码	个人评审
单元测试计划结束时	单元测试计划	个人评审
集成测试开始时	集成测试计划	小组评审

5.5 小 结

保证最终的软件没有故障是项目经理主要关心的问题之一。在质量管理的过程化方法中，先规划质量控制过程，然后正确地执行它们。在定量化质量管理方法中，要设定项目的定量化质量目标；为了实现这个目标，就要定量地监督过程的执行。

下面给出了 Infosys 公司通过故障预防实现定量化质量管理方法的主要经验

教训：

- 与工作量和进度的管理一样，你可以通过使用故障数作为质量的指标对质量进行管理。
- 根据验收测试阶段的故障数设定项目的质量目标。使用以往关于过程能力的数据设置此目标。
- 使用以往的数据，估计过程的各故障检测阶段的故障数。将这些估计的故障数与项目执行时发现的实际故障数相比较，查看项目是否令人满意地朝着实现目标的方向发展，是否需要采取某些纠正措施。
- 除了测试外，还要对评审进行规划，明确地规定评审时机、评审项和评审类型。
- 如果项目的质量目标比以往的绩效更高，则采用与以往的项目相同的过程不能实现当前的质量目标。为了实现这一更高的质量目标，必须对过程进行增强。
- 使用故障预防作为实现项目的更高质量和生产率目标的策略。对于故障预防，确定故障预防团队、执行故障分析的时机以及期望的效益。

本章描述的方法满足 CMM-3 的软件产品工程（Software Product Engineering）KPA 的质量规划需求和互相评审 KPA 的规划需求。它们还满足 CMM-4 的软件质量管理 KPA 的定量化质量规划需求。故障预防规划满足 CMM-5 的故障预防 KPA 的某些需求。

5.6 参考文献

[1] International Standards Organization. *Information Technology — Software Product Evaluation – Quality Characteristics and Guidelines for Their Use.* ISO/IEC IS 9126, Geneva, 1991.

[2] N.E. Fenton and S.L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach,* second edition. International Thomson Computer Press, 1996.

[3] B. Boehm. *Software Engineering Economics.* Prentice Hall, 1981.

[4] A.L. Goel. Software reliability models: Assumptions, limitations and applicability. *IEEE Transactions on Software Engineering,* 11, 1985

[5] S. H. Kan. *Metrics and Models in Software Quality Engineering.* Addison-Wesley, 1995.

[6] J. D. Musa, A. Iannino, and K. Okumoto. *Software Reliability: Measurement,*

Prediction, Application. McGraw Hill, 1987.

[7] L. H. Putnam and W. Myers. *Measures for Excellence: Reliable Software on Time, within Budget*. Yourdon Press, 1992.

[8] L. H. Putnam and W. Myers. *Industrial Strength Software: Effective Management Using Measurement*. IEEE Computer Society Press, 1997.

[9] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys*. Addison-Wesley, 2000.

第6章 风险管理

软件项目是一项复杂的任务。很多不可预见的事件会对项目的成本、进度或者质量产生负面影响。风险管理（risk management）试图使由于意外事件而导致项目失败的概率降到最小。风险管理的目标不是避开研究有风险的项目，而是使风险对正在从事的项目的影响最小。拿 Infosys 公司的 CEO 奈良亚那·墨西（Narayana Murthy）的话来讲："任何值得一做的事情都有风险，领导者面临的问题不是避开它们，而是通过解除风险的策略有效地管理它们。"不正确的风险管理是很多项目失败的根源，主要是由于过于乐观的通病造成的。

瓦苏（Vasu）被任命为一个著名的跨国公司所承担的一个大型项目的项目经理。该项目的任务是构建全球人力资源综合管理系统的多个部件。对于最终系统，瓦苏团队准备要开发的软件必须与另一个软件商开发的系统相集成。客户提供了专门供项目使用的工具，不久以后该工具就发行了新版本。瓦苏团队有 35 人，他们在 1 年多点时间内交付了软件。尽管项目雇佣了一个良好的团队，并且项目经理作出了合理的估计，但是该系统晚了 6 个月才正式投入使用，Infosys 公司支付了项目 50%工作量的蔓延。

为什么该项目会失败呢？有两个原因是很明显的。首先，另一个软件商开发的软件没有按时完成，并且提供给瓦苏团队的接口不断改变。第二，开发期间客户发行了一个新版工具，并要求软件使用这个新版工具。

这两个事件显然都是没有得到正确管理的风险实例。这些风险在开始时是显而易见的，然而，如同任何一个风险一样，它们是否会成为实际的危险却是不确定的。项目开始时，项目经理、业务经理和客户都希望那个软件商会按时交付软件，并且新版工具不会对项目产生影响。

事后，瓦苏认识到如果当时成立了一个指导小组（由两个项目的项目经理和一个客户代表组成），以正确地协调这两个项目及其软件交付时间，很可能会使延迟时间最短。对于第 2 个风险，他认为软件应先用旧版工具进行开发，然后通过一个不同的项目将它移植到新版工具上。解决第一个风险的方案所需的额外工作量最小，并且客户的成本的微不足道。解决第二个风险的方案明显地要客户付出额外的成本。也许为了避免客户感到不满意，所以没有强调该风险，也没有规划

缓和该风险的措施。

本章先讨论风险和风险管理的基本概念，然后讨论 Infosys 公司进行风险评估和风险控制的方法——这是两个主要的风险管理步骤。

6.1 风险和风险管理概念

风险是那些可能发生的事件或者条件，如果它确实发生了，则它的发生会对项目产生有害的或者负面的影响。风险不应与需要管理层干预或者行动的事件和条件相混淆。项目经理必须处理和规划那些有可能发生但是其确切性质却不能预先知道的情况；然而，这种情况不是风险。例如，几乎在软件测试期间总是会发现故障，因此一个合理的项目必须作好发现故障时对它们进行修复的计划。同样，几乎总是会出现某些变更申请，因此项目管理必须相应地准备好变更和计划，以处理这些正常的事件。

另一方面，风险是一种概率事件——它可能发生也可能不会发生。因此，我们通常会表现出很乐观，不是看不到风险就是希望它们不会发生。社会与组织因素也可能会对风险产生误解，并失去明确地标识出风险的勇气。[1]如果风险真的出现了，这种态度会使项目陷入困境，这是一个大型项目中很可能发生的事情。因此，风险管理被认为是管理大型软件项目的最佳实践之最。[2]

风险管理旨在识别出风险，然后采取措施使它们对项目的影响最小。风险管理是软件管理中相对较新的领域。它首次出现于勃姆（Boehm）关于风险管理的指南中。自那以后，有几本书谈到了软件的风险管理。[4,5]

在我们讨论软件环境中的风险之前，让我们用一个例子来进一步辅助说明风险这个概念。考虑一个计算机展览会，其关键目标是提供不间断的计算机服务。为了达到此目的，一个明显的风险是电源故障。电源可能发生故障，也可能不会发生故障。如果它确实发生了故障，即便只断电 1 秒钟，它对计算机服务的影响也是巨大的（机器不得不重新启动，数据可能被丢失等等）。如果这种情况是不能接受的(即，如果电源断电的成本很高)，则可以部署一个 UPS(uninterrupted power supply，不间断电源）使其后果最小。如果怀疑电源会长时间断电，则必须准备一个备用的发电机使这一问题最小化。有了这些风险管理系统后，在电源断电的情况下，即便是长时间断电，则与展览相关的目标也不会受到影响。

从这个例子中应当认识到的第一件事情是：风险管理需要付出额外的成本。

在这个例子中，UPS 和发电机的成本是额外的，因为在不存在电源故障的情况下这些设备是不需要的（例如，如果电力公司保证不会断电）。因此，只有在风险管理的成本大大低于风险真正发生时招致的损失的情况下，才认为风险管理是划算的。[5]（实际上，风险管理的成本应当小于损失的期望值（expected value），稍后将定义期望值概念。）例如，如果断电的损失很小，则 UPS 的成本是不划算的——例如，家庭照明就不用 UPS。

其次，风险管理的价值不容易度量，尤其是事后度量。如果在展览期间电源断电达半个小时，则 UPS 和发电机提供的价值可以根据电源断电情况下使计算机继续运转而实现的“补偿”进行计算。然而，假设电源不会断电 1 秒钟，因此 UPS 和发电机没有使用。这是否意味着这些设备的开销是一种浪费呢？不是，因为电源可能已经断过电。如果风险没有真正发生，则风险管理的价值不能直接根据产生的价值或者结果来度量。因为风险事件很可能不会经常发生，所以在项目期间不用风险管理系统的可能性很高。正是由于风险的这种概率特性及其并非总能实现风险缓和工作的价值，使得风险管理很困难。

从这个例子可以看出，显然风险管理的第一步是识别出可能的风险（本例为电源故障），并评价可能出现的后果（丢面子或者失去客户）。在完成了风险评估以后，你就可以开发一个风险管理计划（例如，配备一个 UPS）。总之，风险管理牵涉到两个关键问题：风险评估和风险控制。每个问题都涉及不同的任务，如图 6.1 所示。[3]

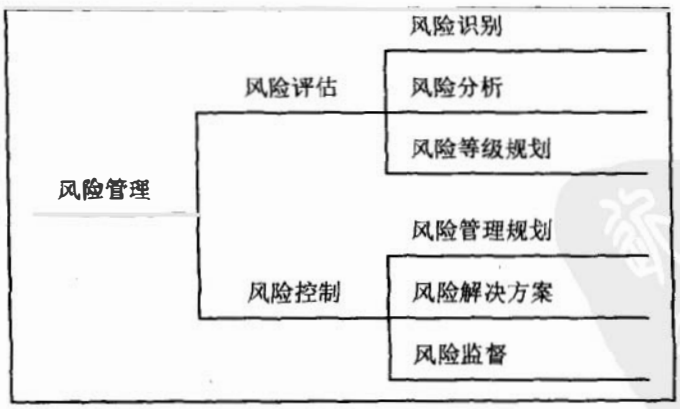


图 6.1 风险管理任务

风险评估任务的目的是识别风险、对它们进行分析，然后划分它们的等级。

在划分风险的等级时，识别出应当进行管理的风险。换句话说，优先等级确定了应当将风险管理的额外工作放在哪里才能使效益最大。对此有两个重要因素。一个是风险发生的可能性；发生的可能性较大的风险自然是风险管理的候选目标。另一个是风险的影响；影响非常大的风险也是风险管理合适的候选。

因此，一种划分风险等级的方法是估计其发生的概率以及发生后造成的后果。这两个值的乘积就是风险损失的期望值，可以用于划分风险的等级。这个期望值称为风险暴露度（risk explosure，简称 RE）。假设 Prob(R)是风险 R 发生的概率，Loss(R)是风险 R 发生时造成的总损失，则风险 R 的风险暴露度 RE 由如下公式给出：[3]

$$RE(R) = Prob(R) \times Loss(R)$$

一旦划分了风险的等级，就必须决定对它们采取什么措施。对哪些风险进行管理是一个管理决策问题，也许只需处理项目中优先级最高的几个风险。

一种方法是采取预防措施或者回避措施，使观察到的风险不再成为一个风险。例如，如果新硬件是一个风险，则用可靠的硬件实现项目就可以避免它了。然而，这种措施并非总能奏效的——例如，如果使用新硬件是客户的一个要求。在这种情况下，必须正确地处理项目的风险。

对于要处理的每个风险，必须先设计风险管理计划，然后再行处理。因为风险认识随时间而变化，所以你还必须对风险及其管理计划的执行进行监督，使风险造成的后果最小。在一个项目中，风险认识可以自然地发展下去，否则风险管理计划也可以采取措施减少风险。在任何一种情况下，必须连续不断地测量风险及其管理计划的状态。

风险管理本身可以集成到开发过程中，软件开发的螺旋模型就是这么做的。[6]如果将风险管理看成一个独立的过程，则必须理解其与项目执行的关系，如图 6.2 所示。如图所示，风险评估和监督利用项目执行的信息以及其他因素来确定要管理的风险。另一方面，风险管理任务使风险的后果最小的同时也将影响项目的过程。

本章后面的内容描述 Infosys 公司的项目经理如何用简单、有效的技术管理风险。风险管理任务可以综合为两个任务：风险评估和风险控制。我们将分别讨论每一个任务。

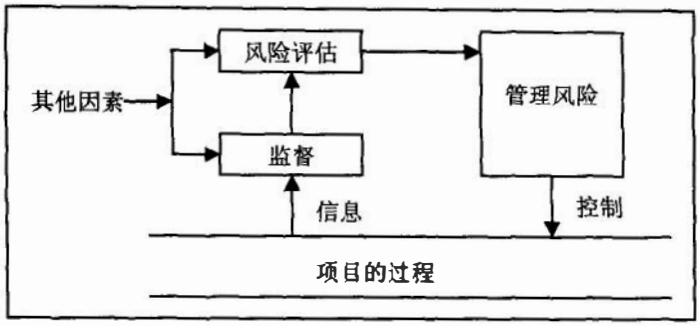


图 6.2 风险管理与项目执行

6.2 风险评估

在 Infosys 公司承担的一个项目中，风险评估任务包括两个传统的组成部分：风险识别和风险等级划分。风险识别任务主要列举出项目可能存在的风险，基本任务是尽量想象出可能使项目出现差错的所有情况。风险等级划分任务考虑所有风险的所有特征，然后划分它们的等级（为了风险管理）。尽管这是两个不同的任务，但是往往同时执行它们。即，项目经理可以在识别出风险的同时对它进行分析。

6.2.1 风险识别

对于一个项目来说，任何可能发生并危及项目成功的条件、情况或者事件都构成一个风险。因此，识别风险就是设想出那些会出现差错的任务。有助于风险识别的方法包括可能存在的风险的检查表、调查表、会议和自由讨论，以及计划、过程和工作产品的评审。[5]经常发生的风险的检查表可能是最常见的风险识别工具。SEI 还对风险进行了分类以帮助风险识别。

Infosys 公司根据对以往项目的调查，汇编了经常发生的项目风险表。该表成为了当前项目识别风险的起点。通常，当前项目的风险可能在列表上。

项目经理也可以用过程数据库获得有关同类项目的风险和风险管理信息。评估和考虑以前遇到过的风险也有助于识别出其他对项目有害的但是在表中没有列出的风险。

项目经理也可以根据他们的判断和经验对情况进行评估以识别出潜在的风险。另外也可以根据项目管理计划的评审和讨论会来引出其他人员对风险的观点。

6.2.2 风险等级划分

已识别出的项目风险只是给出了有可能阻碍项目实现目标的事件。但是各种风险的后果不可能是相同的。在你继续进行风险管理之前，必须划分它们的等级以便管理精力能够集中在最高等级的风险上。

风险等级划分要求分析风险事件真正发生时可能产生的影响。即，如果风险真的发生了，项目会受到什么损失呢？损失包括直接损失、由于失去商业机遇或者未来的业务而造成的损失、由于雇员信心下降而造成的损失等等。根据可能造成的后果和风险事件发生的概率，可以计算风险暴露度，然后就可以用它划分风险等级。

该方法要求定量化评估风险概率和风险后果。通常，能够帮助你定量化估计这些参数的历史数据很少。因为风险是概率事件，它们并不经常发生，因此很难收集到关于它们的数据。此外，必须正确地解释任何这样的数据，因为管理风险的行为对它们有影响。这一事实表明风险等级划分将基于经验，而不是基于以往的硬数据。在这种情况下，对风险发生的概率和可能造成的后果进行分类有助于区分高等级的风险和低等级的风险。[5]在 Infosys 公司，风险发生的概率分成低、中和高三个等级。表 6.1 给出了每种风险类别的概率范围。

表 6.1 风险分类

概率	范围
低	0.0-0.3
中	0.3-0.7
高	0.7-1.0

为了评定某个风险对项目的影响，必须选择一个影响单位。为了简化风险管理，Infosys 公司的项目经理用 1 到 10 之间的数值来衡量风险影响。在该数值范围内，风险影响可以被评价为低、中、高和非常高。表 6.2 给出了这些等级中每个等级的影响范围。

表 6.2 影响分类

等级	范围
低	0.0-3.0
中	3.0-7.0
高	7.0-9.0
非常高	9.0-1.0

有了这些等级以及每个等级的范围之后，就可以规定如下简单的风险等级划分方法：

（1）对于每个风险，将其发生的概率表示成低、中、高。如果需要，为每个等级分配给定范围内的概率值。

（2）对于每个风险，用低、中、高或者非常高评价其对项目的影响。如果需要，分配一个从 1 到 10 的权重。

（3）基于风险发生概率及其对项目的影响，划分风险的等级；例如，一个发生概率为高、影响为高的风险条目的风险等级，与一个发生概率为中、影响为高的风险条目的风险等级比，显然前者高于后者。在出现冲突的情况下，充分利用你的判断力（或者分配数字以计算风险暴露度的数值）。

（4）选择几个风险等级最高的风险条目，缓和它们的风险并对它们进行跟踪。

风险管理的主要目标是识别出几个最危险的风险条目，然后集中解决它们。因此，使用分类方法可以起到良好的效果。显然，一个发生概率为高和影响等级为高的风险是一个具有高风险暴露度的风险，因此具有较高的风险管理等级。

在你对风险进行分类时，如果风险概率和风险影响等级是（高，中）或者（中，高）时会出现一个等级划分难题。在这种情况下，哪个风险应当被评价为更高的等级是不明确的。在这种情况下，简单的处理方法是同时缓和这两种风险。如果需要，可以用实际数值区分这些类型的风险等级。

这种划分风险等级的方法有助于将重点放在高风险的条目上，但是它不能帮你分析哪种风险缓和策略的成本效益比最好。即，这种方法根据一个标量来说明风险影响，而不用货币价值，所以它不支持你根据货币来计算期望的损失。因此，你不能分析某种需要付出一定成本的风险缓和策略是否值得采用。然而，这种分析一般情况下是不需要的，因为风险管理的重点往往在于以最低的成本管理风险，而不在于风险管理本身是否有益。当然，如果你必须决定是否应当管理一个风险，或者不对它进行管理是否是经济上最明智的选择，则你必须知道风险的经济影响。

6.3 风险控制

在项目经理识别出风险并划分它们的风险等级之后，问题就变成了如何处理它们。只有在你能够准备一个风险缓和计划使风险的后果最小的情况下，知道了风险才有意义——这是风险管理的基本目标。风险管理的第 2 步：风险控制，它将使风险的影响最小。本质上，这一步牵涉到制定缓和风险的计划，然后执行计划和监督风险。

6.3.1 风险管理规划

在识别出风险并划分了它们的风险等级之后，项目经理应当处理哪些风险就很清楚了。为了管理风险，正确的规划是必不可少的。风险管理规划的主要任务是确定使风险后果最小所需的措施，通常称为风险缓和措施（risk mitigation step）。与风险识别一样，你可以查阅一个常用的风险缓和措施表，并针对每个风险选择一种合适的风险缓和措施。Infosys 公司所用的风险缓和措施表如表 6.3 所示。该表不仅是识别风险的起点，也是在划分风险等级后选择风险缓和措施的起点。与风险识别一样，风险缓和措施并不局限于表 6.3 提到的措施。你也可以使用过程数据库来识别风险和选择风险缓和措施。

表 6.3 中的大多数风险和缓和措施都是显而易见的。从中可以看出，前面几个风险都与人力和需求相关。这里的许多条目与勃姆[3]和霍尔（Hall）[5]给出的最危险的风险表中的那些条目相类似。选择一个风险缓和措施不仅仅是一种智力任务，还必须执行（和监督）风险缓和措施。为了确保正确地执行了所需的措施，必须在项目的进度计划中提到它们。换句话说，你必须更新项目的进度计划（它列出各种任务，并规定了它们将在什么时候执行），使它包含与所选的风险缓和措施相关的任务。

6.3.2 风险监督和跟踪

风险等级划分以及随后的规划都是基于执行风险分析时对风险的认识。第一次风险分析在项目规划期间发生，而初步的风险管理计划反映了那时对情况的看法。因为风险是概率事件，通常依赖于外部因素，所以由风险引起的危险可以因因素的改变而随时间发生变化。显然，风险认识也可以随时间而变化。此外，风险缓和措施可能会影响风险认识。

表 6.3 前 10 个风险及其风险缓和措施

序号	风险类别	风险缓和措施
1	缺乏经过技术培训的人力	● 估计可以留出初步学习时间的人数 ● 保留额外资源的候补人员 ● 确定项目特有的培训程序 ● 举行交叉培训会议
2	需求变更太多	● 得到客户保证不再修改初始需求规格说明 ● 使客户相信需求变更将影响进度 ● 定义一个处理需求变更的程序 ● 协商实际工作量的支付
3	需求不明确	● 根据经验和逻辑推理作出某些假设，并及时通知客户；保证客户不再提出不明确的需求 ● 开发一个原型，并让客户评审需求
4	人员流失	● 确保在关键的项目领域分配多重资源 ● 举行团队建立会议 ● 团队成员间岗位轮换 ● 保留项目的额外资源以备后用 ● 用文档正确地记录每个成员的工作 ● 严格遵守配置管理过程和指南
5	外部强加给项目的决策	● 用有力的事实和数据概括出不利条件，并与负责强制执行决策的人员协商 ● 如果必须执行强加的决策，识别出实际风险并遵循其缓和计划
6	没有满足绩效需求	● 明确定义绩效指标，并让客户评审它们 ● 定义满足绩效指标的标准，并遵循它 ● 准备满足绩效指标的设计，并评审它 ● 模仿重要事务的绩效或者确定绩效模型 ● 只要有可能用一些有代表性的数据进行测试 ● 只要有可能执行强度测试
7	不切实际的进度	● 通过协商确定一个更好的进度计划 ● 识别出并行任务 ● 提前准备好资源 ● 识别出可以自动操作的领域 ● 如果进度计划没有包含关键路径，则与客户协商 ● 协商实际工作量的支付
8	运用新技术（硬件和软件）	● 考虑按阶段提交 ● 先提交关键模块 ● 在进度计划中包含学习时间 ● 提供新技术培训 ● 开发一个概念证明（proof-of-concept）应用程序
9	业务知识不够	● 增强与客户的交流，确保从客户那里得到足够的知识 ● 组织领域知识培训 ● 模仿或者提出商业事务的原型，并得到客户的认可
10	链路故障或者绩效低下	● 正确估计客户的期望 ● 提前计划链路负荷 ● 计划最优的链路使用

动态主义者认为不应当把项目中的风险看成静态，而是必须定期对它们进行重新评估。因此，除了监督计划的风险缓和措施的进展外，还必须定期重新访问整个项目的风险认识。这种评审的结果在每次里程碑分析报告中进行报告（参见第 11 章）：报告风险缓和措施的状态以及当前的风险认识和策略。为了准备这个报告，必须重新进行风险分析以确定风险等级是否已经发生变化。

6.4 实 例

本节包括两个风险管理计划：一个是 ACIC 项目的风险计划，另一个是另一个项目的计划（这里称为 XYZ）。正如你将会看到的，风险管理计划都比较小——通常是一页纸上的一个表。缓和计划中提到的任务成为项目任务的一部分，并且有可能显式地调度它们。

6.4.1 ACIC 项目

ACIC 项目经理选择用数字进行风险等级划分和分析。如表 6.4 所示，前面几个风险条目具有从 8 到 3 的影响等级，同时还要显示每个风险的缓和措施。

表 6.4 ACIC 项目的风险管理计划

序号	风险	概率	影响	风险暴露度	风险缓和计划
1	我们可能需要数据库设计师和客户的数据库管理员的支持	0.5	8	4	仔细地规划每组任务所需的时间，并事先给予充分的注意
2	因为第一次使用 RUP，所以团队对它的理解可能不全面	0.9	3	2.7	与 Infosys R&D 实验室的专家密切合作 在整个项目中保持与客户交流并提交任何进度或者工作量偏差 向团队培训 RUP 方法
3	人员流失：团队成员可能临时离开	0.3	7	2.1	分配任务时使多个人了解项目中的多个单元和用例
4	通过链路与客户的大型机 DB2 通信：链路可能不如期望的那样那么有效	0.1	8	0.8	执行额外的代码评审、桌面检查等以使对链路的依赖最小 链路绩效下降时立即提交

例如，第 2 个风险使用了 RUP 方法。即，项目引入了一个风险，因为这种方法对于项目人员来说是一种新方法。此外，似乎其他项目还没有用过 RUP。其中

一种风险缓和计划是最明显的缓和措施：制定RUP方法的培训计划。此外，缓和计划建议咨询R&D实验室的人员，因为他们掌握了RUP及相关概念的知识，必须持续不断地与客户进行交流，并且及时提交任何问题。

注意，一旦这些计划作为风险缓和措施而被接受了，它们将影响项目的详细进度计划；进度计划必须包含相应的培训和概念证明（proof-of-concept）构建任务的时间。这些需要将随着很多风险缓和措施而出现。因为它们代表了行动，并且因为详细的项目进度计划表示项目要采取的大部分措施，所以风险缓和措施通常会改变详细的进度计划，增加项目的总工作量需求。

6.4.2 XYZ项目

该项目用等级系统进行风险管理。表6.5给出了各种等级评定方法和风险缓和措施。此风险管理计划是项目管理计划的一部分，也是从它那里提取出来的。

表6.5 ZYX项目的风险管理计划

序号	风险	概率	影响	风险暴露度	缓和计划
1	不能满足高绩效	高	高	高	● 通过需求原型，向客户指明期望的绩效 ● 用知识体数据库中的提示改进绩效 ● 使团队注意需求 ● 更新评审检查表以找出绩效缺陷 ● 不断地研究和改进绩效 ● 遵循早期绩效研究的指南 ● 集成和系统测试阶段，测试应用满足绩效期望
2	缺乏具有正确技能的人员	中	中	中	● 人员培训 ● 与客户一起评审原型 ● 制定编码实践
3	应用需求的复杂性	中	中	中	● 确保知识转化 ● 分配具有该应用程序经验的人员
4	人员流失	中	中	中	● 培训一个由4人组成的核心小组 ● 团队成员间岗位轮换 ● 确定关键角色的候选人才
5	需求不明确	中	中	中	● 评审原型 ● 执行里程碑评审
6	在脱机开发期间协调联机维护时作出的变更的难度	中	低	中	● 制定管理计划并遵循良好定义的协调方法 ● 每月协调一次（第1个星期的星期二或者下一个工作日） ● 不要协调某个截止日期后的变更

在里程碑处执行风险分析的方法基本上与前面所述的一样，但是将更加注意项目计划中列出的风险（即，更加强调项目早期风险分析的结果）。风险分析期间，项目经理可以划分风险等级。在 XYZ 项目中，大约在制定初步风险管理计划后 3 个月进行里程碑分析时，风险认识已经发生了某些变化。表 6.6 给出了一些暴露度发生了变化的风险。

表 6.6 XYZ 项目的风险变化

序号	风险	当前概率	当前结果	当前风险暴露度
2	人员流失	高	高	高
3	在脱机开发期间协调联机维护时作出的变更的难度	低	低/中等	低

基于迄今具有的项目经验，项目经理认为变更协调的结果还不算太差。这种情况是有可能出现的，因为协调问题比预想的要简单多了。同样，人员流失的风险认识已提高了（可能是由于团队成员的经验），而团队成员在项目中期离去的事实，现在与项目开始时比被看成一个更大的问题。每当分析风险时，根据项目的当前事实和风险的性质，风险缓和计划也可能发生变化。在这个项目中，风险缓和计划没有改变。

6.5 小 结

项目风险是这样一种条件，其发生是不确定的，但可以对项目产生负面影响。风险管理要求识别出风险，划分它们的等级，并对几个最主要的风险采取缓和措施使它们的影响最小。在风险没有实际发生时，风险缓和的成本看起来好象是一种浪费，但是必须引入它们以使风险真正发生时的损失最小。

下面列出了从 Infosys 公司的风险管理方法中吸取的关键教训：

- 为了帮助你识别出风险，可以用一个经常发生的风险列表作为风险管理的起点。此外，预先作好准备并设想出项目中可能出错的任何东西。
- 对于风险等级划分，一种简单而有效的机制是将风险的概率以及它们的影响按低、中、高几个类别进行分类，然后管理具有高概率和高影响的风险。
- 对于几个最主要的风险，计划风险缓和措施，并确保项目期间正确地执行它们。

- 监督风险并定期重新评估风险（也许在里程碑处），检查风险缓和措施是否有作用，并重新评估风险认识。

关于 CMM，CMM-2 的项目规划 KPA 要求一个项目有一个风险管理计划。正确的风险管理和监督过程是 CMM-3 的集成软件管理 KPA 的一个需求。

6.6 参考文献

[1] R. N. Charette. Large-scall project management is risk management. *IEEE Software*, July 1996.

[2] N. Brown. Industrial-Strength Management strategies. *IEEE Software*, July 1996.

[3] B. Boehm. *Tutorial: Software Risk Management. IEEE ComputerSociety*, 1989.

[4] R. Charette. *Software Engineering Risk Analysis and Management.* McGraw Hill, 1989.

[5] E. M. Hall. *Managing Risk: Methods for Software System Development.* Addison-Wesley, 1998.（"SEI 软件工程丛书"之《风险管理》，伊莱恩·M·霍尔著，其英文影印版和中译版均由清华大学出版社于 2002 年出版。）

[6] B. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, May 1988.

[7] M. Carr et al. *Taxonomy-based Risk Identification.* Technical Report, CMU/SEI-93-TR-006, 1993.

第 7 章　计划的度量和跟踪

项目管理计划仅仅是一个可以用来指导项目执行的文档。除非根据计划对项目执行的实际绩效进行跟踪，否则计划所起的作用是有限的。而对于项目跟踪，某些关键参数的值必须在项目执行期间得到度量。跟踪不是一项轻松的任务，并且与其他任务一样，如果你要正确地执行它，你必须制定跟踪计划。制定跟踪计划时，你必须确定如何跟踪任务、工作量和故障等问题、使用什么工具、遵循什么报告结构以及多长时间报告一次，等等。

本章讨论 Infosys 公司的项目如何进行度量，同时还描述了项目跟踪规划和绩效变化阈值的选择，阈值用于触发管理行为。实际的项目跟踪将在第 10 和 11 章中讨论。

7.1　度量概念

项目度量的基本目的是有效地控制项目。本节讨论与指标和度量有关的概念，以及控制一个项目时应当度量的基本指标。过程控制的一种方法是统计过程控制。本节还讨论了一些与 SPC 相关的概念以及 SPC 在软件中的使用方法。

7.1.1　指标与度量

软件指标可用于定量表示软件过程或者软件产品的各种特征。过程指标使软件过程或者开发环境定量化，而产品指标是软件产品的度量（1，2）。产品指标独立于生产产品所用的过程。例如，生产率、质量、资源指标、故障引入率以及故障排除效率就是过程指标；而产品的规模、可靠性、质量（质量既可以看成产品指标，也可以看成过程指标）、代码复杂性和功能就是产品指标。

要使用指标必然要求制定度量方法以获取数据。对于任何一个指标方案，你必须明确地理解收集数据的目标以及基于这些数据的判断模型。通常，使用哪些指标及采取哪些度量方法将取决于项目和组织目标；你可以使用一个框架（诸如目标-问题-指标模式）来确定需要度量的指标。[3,4]然而，实际上大多数情况只

要几个指标就足够了，只有在特殊情况下才需要特殊的指标。进度、规模、工作量和故障都是项目的基本度量，它们形成了一个稳定的指标集。[5,6]

进度是最重要的指标之一，因为大多数项目都是在进度和最后期限驱使下进行的。然而，进度是最容易度量的，因为它通常使用日历时间。工作量是软件项目消耗的主要资源。因此，工作量跟踪是监督项目执行时的关键任务；它对于评价项目是否在预算内执行很重要。即，工作量数据是声明诸如“项目的成本可能比以前的项目高出30%左右”或者“项目有可能在预算内完成”所需的。

因为故障直接关系到项目质量，所以故障跟踪对于确保质量很重要。一个大型软件项目可能包含上千个故障，它们由不同的人在不同阶段发现。通常修复故障的人并不是发现故障或者报告故障的人。通常，项目经理希望在最后发行软件之前排除大部分或者全部故障。在这种情况下，故障报告和结论不能用非形式化的方法完成。使用非形式化的方法会导致现在发现的故障后来又忘记了，因此故障不但没有被排除而且还花额外的时间去再次发现它们。因此，至少必须记录故障并跟踪它们的排除。对于这个过程，你需要信息，诸如故障的表示、故障的位置、发现故障的人的名字和排除故障的人的名字。只要记录了发现的每一个故障（随后再排除它），就可以重点分析迄今已发现了多少个故障、百分之多少的故障还没有排除以及其他一些问题。故障跟踪被认为是管理一个项目的最佳实践之一。

仅仅记录故障并对它们进行跟踪并不足以支持其他必不可少的分析。为了知道在哪里发现了百分之多少故障，你还需要记录发现故障阶段的信息。为了理解各质量控制任务的故障排除效率并因此改进它们的绩效，不仅必须知道故障是在哪里发现的，而且还要知道是在哪里引入的。换句话说，对于记录的每个故障，还应当提供关于引入故障阶段的信息。

项目规模是另一个基本指标，因为很多数据（例如，已交付软件的故障密度）都是根据规模进行标准化。没有规模数据，就不能用以往的数据预测绩效。此外，如果不根据某个标准的规模度量进行标准化，就没有了绩效比较的基准。项目规模的两个常用的度量单位是代码行（lines of code，简称LOC）和功能点。如果使用代码行作为度量单位，生产率会随编程语言而不同。而功能点度量却不会因编程语言而不同。

7.1.2 通过统计过程控制的监督过程

统计过程控制已经在制造业取得了成功应用，并且其在软件领域的应用也在

逐年上升[8]。这里我们简单地讨论 SPC 的一些基本概念，有关的更多信息，你可以参考任何关于统计质量控制的教材。[9,10]在第 10 和 11 章，你将会看到 SPC 概念如何用于项目监督。

过程用于生产产品，而产品的质量可以根据某些质量特征进行定义。很多因素影响这些特征值的可变性。这些因素可以分成两类：可变性的自然（即内在）原因和指定的（即特殊）原因。自然原因是那些总是存在的原因，并且其中的每一个都对可变性有影响。除非过程本身发生了变化，否则控制这些因素是不切实际的。而指定的原因是那些偶尔出现的原因，对过程绩效的可变性有很大的影响，并且可以控制它们。图 7.1 说明了这些原因和质量特征间的联系。

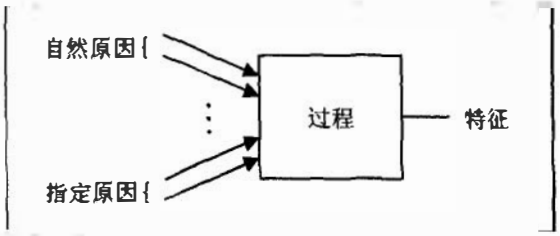


图 7.1　指定原因和自然原因

如果质量特征的可变性仅仅由于自然原因引起的，则认为过程在统计控制下。SPC 的目标是保持生产过程在统计控制下。

控制图（control chart）是一种应用 SPC 的工具。为了建立一幅控制图，将一个过程的结果看成一个代表感兴趣的特征值的数值流。子组数据取自这一数值流，并且子组数据的平均值在图上绘出，形成一幅 X-柱状统计图。还要确定控制下限（lower control limit，简称 LCL）和控制上限（upper control limit，简称 UCL）。如果点落在控制极限外面，则认为这种大的变化是由于指定的原因造成的。还有一幅图称为 R 图，绘出所选子组数据的范围（最小值与最大值的差）。在 R 图的控制极限已知的情况下，落在这些控制极限外面的点也被认为由指定原因造成的。

根据约定，LCL 和 UCL 通常被设为 3Σ 平均值，其中Σ是仅有正常可变性的数据（即，由自然原因造成的可变性）的标准差。有了这些极限后，一次错误报警（false alarm）（具有自然可变性的点落在极限外）的概率仅为 0.27%。

如果生产过程不能重复地产生相同的项目，就如软件过程那样，形成子组并没有多大意义；因此必须考虑单值。对于这种过程，可以使用 XMR 图。[9,10]在

一幅XMR图中，两个连续值的取值范围被看成R图的范围。对于X柱形图，绘出各单独的值；然后，用平均取值范围确定控制极限。

注意控制极限不同于规范极限。规范极限（specification limit）根据需求规定过程所必须的绩效。而控制极限基于过程的实际数据确定过程的实际绩效——即，过程真正能够达到的绩效。显然，如果控制极限都在规范极限内（规范极限范围大于控制极限），则过程能够在大部分时间内提交满足规范的结果。另一方面，如果规范极限在控制极限内，则过程生产的结果不满足需求的概率增大。根据控制极限和规范极限间的关系，就可以形式化地定义一个过程的能力。

你可以使用控制图连续地监督过程的绩效，并识别出失控的情况。另外，在表示结果的点落在控制极限外的情况下，必须决定采取什么措施。通常可以采用两种类型的措施：

- 对结果进行返工，以便它具有可接受的特征——即，采取纠正措施。
- 执行进一步的分析以识别出指定的原因并从过程中删除它们——即，采取预防措施。

为了利用软件过程的控制图，必须先确定能够应用SPC的过程。一种是选择整个过程，其结果就是最后要提交的软件产品。对于这一过程的结果值得研究的特征包括生产率、已交付软件的故障密度和故障引入率，以及其他一些特征。对于总体过程的大部分结果的特征值只有在项目收尾时才能得到，因此总体过程的SPC对项目监督和控制所起的作用是有限的。它的作用主要在于理解和改进软件过程的能力。

为了控制一个项目，你可以对在项目期间执行的“微型过程”利用SPC，诸如评审过程或者测试过程等。在SPC控制下，过程一执行就可以分析其结果。如果需要，你就可以采取纠正措施或者预防措施实施控制。通过纠正措施，使界限外的结果变成可接受；预防措施有助于改进项目其余任务的执行绩效。第10章和第11章讨论了用SPC监督项目。

给定绩效变化很大的软件过程的绩效变化概率，要确定只有自然可变性的点以便控制极限，这不是一件容易的事情。因此，为了根据以往的绩效数据计算控制极限，你必须使用判断力来确定应当排除哪些点。此外，也不应当盲目地使用以往的数据。有见识的管理层必须始终支持使用以往数据。例如，不能仅仅因为绩效不在根据以往数据计算的范围之内，就认为过程已经失败了。[2,11]一种更加合理的方法是用绩效范围引起对绩效偏差的注意，然后分析引起偏差的原因。

7.2 度　　量

对一个项目的任何定量控制主要取决于项目执行期间的度量。为了在项目期间执行度量，你必须仔细地规划度量什么、何时度量以及如何度量。因此，度量的规划是项目规划的一个关键要素。本节讨论 Infosys 公司的项目所执行的标准度量方法。项目经理可以在他们的项目需要时增加这些度量。

7.2.1 工作量数据收集

为了帮助项目经理监督工作量，每个雇员用一个周任务报告（weekly activity report，WAR）系统记录各种任务所花的工作量。这是内部开发的一个联机系统，所有提交的WAR 都保存在一个中心数据库中。每人每周提交其 WAR。提交时，报告送到各人的监督人那里以得到批准。批准之后，WAR 提交就结束了，并且不能改变。每个人都提交一个 WAR，包括 CEO 在内。如果某人在给定的时间内没有提交 WAR，则认为他休假了。

一个 WAR 记录项由一个记录序列组成，每周 1 个记录。每个记录是一个条目列表，每个条目包含如下字段：

- 程序代码
- 模块代码
- 任务代码
- 任务描述
- 从周一到周日的工作小时数

任务代码表示任务类型的特征。程序代码和模块代码的工作量数据根据模块或者程序进行划分，这在考虑组件级监督是重要的。为了支持分析和项目比较，根据报告的工作量对任务进行标准化是很重要的。拥有一个标准化的任务代码集有助于实现此目标。表 7.1 给出了 Infosys 公司的项目采用的任务代码。（这些代码不同于我前本书中给出的那些代码，因为 Infosys 公司在引入新的基于 web 的 WAR 系统后代码发生了变化。）

表 7.1 工作量的任务代码

任务代码	描述	任务代码	描述
PAC	验收	PIA	影响分析
PACRW	验收测试后的返工	PINS	安装/客户培训
PCAL	项目捕获器	PIT	集成测试
PCD	编码和自我单元测试	PITRW	集成测试后的返工
PCDRV	代码走查/评审	PRI	项目启动
PCDRW	代码走查后返工	PPMCL	项目收尾任务
PCM	配置管理	PPMPT	项目规划和跟踪
PCOMM	沟通	PRES	技术问题研究
PCSPT	客户支持任务	PRS	需求规范任务
PDBA	数据库管理任务	PRSRV	需求规范评审
PDD	详细设计	PRSRW	需求评审后返工
PDDRV	详细设计评审	PSP	策略规划任务
PDDR	详细设计评审后的返工	PST	系统测试
PDOC	文档	PSTRW	系统测试后的返工
PERV	模型和图纸的评审	PTRE	项目特有的培训任务
PERW	模型和图纸的返工	PUT	独立的单元测试
PEXEC	建模和制图	PUTRW	独立单元测试后的返工
PHD	概要设计	PWTR	等待资源
PHDRV	概要设计评审	PWY	保修期间的工作量
PHDRW	概要设计评审的返工		

任务代码为很多阶段的返工提供了专门的代码。这种分类有助于计算质量成本。达到这一级精化后，你就可以按阶段分析工作量数据或者按子阶段分析工作量数据。项目规定的程序代码和模块代码用于记录项目的不同单元的工作量数据，从而简化对单元的分析。

为了简化项目级计划的工作量和实际所花工作量的分析，WAR 系统与项目的 MSP（Microsoft Project）描述相连接。项目人员只有在提交了项目的 MSP 后才开始提交项目的 WAR（一旦提交了 MSP，系统就知道了哪些人应该从事该项目）。计划的任务被定义成那些在 MSP 中列出的任务，并分配给项目中一个被授权的人。没有计划的任务都是其他项目任务。

输入一周的 WAR 时，用户根据屏幕进行操作，屏幕被分成两部分：计划的任务和未计划的任务。在 MSP 中，每周分配给某个特定的人的所有任务都显示在

那个项目的计划的任务部分。用户不能增加或者修改该部分中显示的任务。她只能输入不同的任务每天所花的时间。为了登记没有在计划的部分中列出的任务所花的时间，用户可以在该项目的未计划的部分输入任务代码、其描述以及这些任务每天所花的时间。

7.2.2　故障记录和跟踪

在 Infosys 公司的一个项目中，故障检测和排除任务如下进行。故障提交者发现并记录故障，这时故障就处于“已提交”状态。接着，项目经理将修复故障的工作分配给某个人，通常是那个存在故障的文档或者代码的作者。这个人执行调试并修复报告的故障，于是故障就进入了“已修复”状态。已修复的故障仍然不能算被排除掉了，还要经其他人（通常是提交故障的人）验证故障已经被修复了。验证以后，故障就可以被标志为“已排除”状态。换句话说，一个故障的基本生命期有三个状态：已提交的状态、已修复的状态和已排除的状态（参见图 7.2）。一个没有被排除的故障仍然称为未被排除的故障（open）。

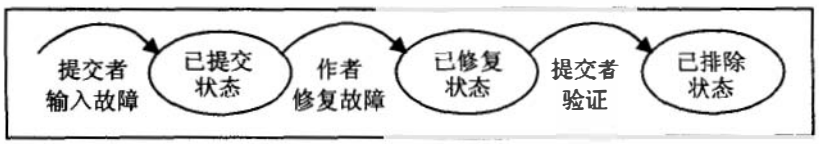


图 7.2　故障的生命期

故障控制系统（defect control system，简称 DCS）在项目中用来记录和跟踪故障。该系统允许进行各种类型的分析。表 7.2 给出了该系统记录的每个故障的的信息。

为了确定故障引入阶段，需要对故障进行分析。虽然故障检测阶段包括评审和测试任务，但是故障引入阶段包括生产工作产品的阶段，诸如设计和编码等。基于故障的性质，可以判断故障是什么时候引进的。故障引入阶段与故障检测阶段不一样，故障检测阶段是确定的，而故障引入阶段却更加不明确；需要根据故障的性质和其他相关信息进行估计。使用引入阶段和检测阶段信息，就可以计算故障排除效率、百分比分配以及其他指标。

表 7.2 故障数据记录

字段名	描述	必须（M）/可选（O）
项目代码	其中发现了故障的项目的代码	M
描述	故障的描述	M
模块代码	模块代码	O
程序名	其中发现了故障的程序的名字	O
检测阶段	其中检测到故障的阶段	M
引入阶段	其中故障被引入或者故障起源阶段	M
类型	故障分类	M
严重性	故障的严重性	M
评审类型	评审的类型	O
状态	故障的当前状态	M
提交者	检测到故障的人的名字	M
作者	故障制造者的名字	M
提交日期	故障被提交给作者的日期	M
结束日期	被提交的故障被排除的日期	M

有时不希望根据故障阶段来理解故障的性质，而是根据故障分类来理解其性质，这也是必要的，这样一种分类有助于你理解各类故障的分配情况。因此，还要记录故障的类型。表 7.3 给出了可能存在故障类型及其相应的例子。一个项目也可以定义它自己的类型分类。

最后，还要记录故障的严重性。此信息对项目经理很重要。例如，如果一个故障很严重，你很可能要为它安排时间以尽快地修复它。此外，你可以决定在交货时间很紧的情况下暂不修复次要的或者不重要的故障。表 7.4 给出了 Infosys 公司使用的故障严重性分类。

根据此信息进行各种分析是可能的。例如，你可以根据类型、严重性或者模块分解故障；根据模块、严重性或者总故障数绘出故障引入和排除的趋势；确定周故障引入率；确定故障排除效率；确定不同阶段的故障引入率；等等。在第 11 章中你将会看到这些数据在监督质量和预防故障方面的一些应用，同时该章还描述了案例研究中要求的故障数据。

表 7.3　故障类型

故障类型	例子
逻辑	算法使用不正确/不充分；错误的条件、测试实例或者设计文档
标准	编码/文档标准问题，诸如缩进、对齐方式、布局、模块性、注解、繁琐的编码和拼写错误等
冗余代码	在很多程序或者相同程序中使用的某些代码块
用户接口	规定的功能键不能起作用；不正确的菜单导航
绩效	处理速度低下；由于文件规模而引起系统崩溃；内存问题
可重用性	代码不能重用
设计问题	与特定设计相关的问题
内存管理故障	诸如内存信息转储、数组溢出、非法的函数调用、系统挂起或内存溢出
文档故障	文档（诸如项目计划、配置管理计划或者规格说明等）评审时发现的故障
一致性	不能在整个系统中以相同的顺序更新或者删除记录
可跟踪性	不能从程序源代码追溯到规范
可移植性	代码不独立于平台

表 7.4　故障严重性

严重性类型	分类解释
紧急的故障	根据对进度的影响，故障可能非常紧急；或者说它可能是一个中断源——即，它不能使用户继续使用系统
主要故障	在很多程序或者模块中发生过的同类型故障。我们必须纠正任何问题。例如，并非所有程序都遵循编码标准。另外，除一个特定的工作区外，故障使用户不能以正常的方式继续工作下去
次要故障	这个故障已被隔离或者不会妨碍用户继续工作，但是它会引起不便
装饰性故障	一种决不会影响软件产品绩效的故障——例如，美观问题或者消息语法错误

7.2.3　进度度量

进度度量比较简单直接，因为你使用日历时间。详细的任务和进度计划通常包含在 MSP进度计划中，因此在 MSP 中给出了估计的任务日期和任务的持续时间。知道了一个任务的实际日期后，你就可以轻松地确定它的实际持续时间。

7.2.4　规模度量

如果采用自底向上估计技术，则根据不同复杂度的程序数估计规模。尽管这一指标对估计是有用的，但它不支持一种能够在项目间有针对性地进行比较的标

准的生产率定义。如果用代码行（lines of code，简称 LOC）作为规模度量，也会出现同样的问题：生产率随编程语言的不同而不同。为了建立一个基准并根据该基准进行绩效比较，必须对规模度量单位进行标准化，采用一种经标准化的统一的规模度量单位，则可以将功能点作为规模度量单位。

已交付软件的规模一般以 LOC 为度量单位，这可以用常规的编辑器和行计数器计算。在项目已经完成并准备提交时执行这种计算。前文已经讨论过，根据以 LOC 为单位的规模度量，可以用已经公布的转换表计算功能点的规模度量[12]。

7.3 项目跟踪

项目跟踪的主要目标是让项目经理掌握项目执行情况，以便他们能够确定是否需要采取什么措施以确保项目目标得到满足。因为满足既定的项目目标是基本要求，所以必须监督所有能够影响达到目标的项目特征，并且必须制定这种监督任务的计划。在 Infosys 公司，项目经理通常制定如下跟踪计划：

- 任务跟踪
- 故障跟踪
- 问题跟踪

任务跟踪检查哪些计划的任务已经完成。如果任务的粒度很小，则你可以把它看成两种最底层任务的状态之一：没有完成或者全部完成。对于更高层的任务，你可以根据最底层任务的状态以及它们的估计计算高层任务的完成情况。

故障跟踪必须与故障记录同时进行。

问题跟踪确保那些有可能延迟项目的澄清和其他问题不会失去控制。第 11 章说明了 Infosys 公司如何执行这些跟踪任务。规划期间，项目经理规定他们计划采用什么类型的跟踪，以及他们将使用什么工具或者方法。

此外，为了跟踪项目的工作量状态、进度状态和质量状态，项目经理还必须制定如下计划：

- 任务级监督
- 状态报告
- 里程碑报告

任务级监督保证正确地完成每个任务。你可以使用统计过程控制定量地监督任务。基于以往的绩效，你就可以确定某些任务的主要绩效参数的极限。然后，将一个任务的实际绩效与既定的极限相比较。如果绩效不在可接受的极限内，你可能要采取某些措施。在 Infosys 公司，评审和测试是两个采用这种方法的主要任务。

状态报告一般每周准备一次，帮助你估计哪些任务已经发生及哪些任务需要完成。在项目的里程碑处，你要执行一种更加复杂的任务，即定量地检查工作量、进度和故障的实际数据和估计数据。此外，你还要监督风险、培训、评审、客户投诉等等。里程碑分析在控制项目方面有着重要的作用。为了确保里程碑分析的频度足以支持及时干预，在客户要求的里程碑相距太远时，计划内部里程碑，以每 3 到 5 周执行一次里程碑分析。

在里程碑处，你要分析实际的工作量、进度和故障以及估计的工作量、进度和故障。如果两者相差甚远，则需要采取某种纠正措施。为了区分正常偏差和重大偏差，必须规定可接受的偏差范围；为了设定这些极限，可以利用控制图概念。Infosys 公司为工作量和进度偏差（实际值与估计值之间的差）定义了控制极限。这些值最初基于判断和经验，现在都基于以往的数据并且按与其他控制极限相同的方式进行计算。Infosys 公司以前的工作量偏差的极限是 35%，而进度极限是 15%；随着过程的改进，这些极限已减少到 20%和 10%。图 7.3 和 7.4 给出了进度和工作量偏差的控制图。

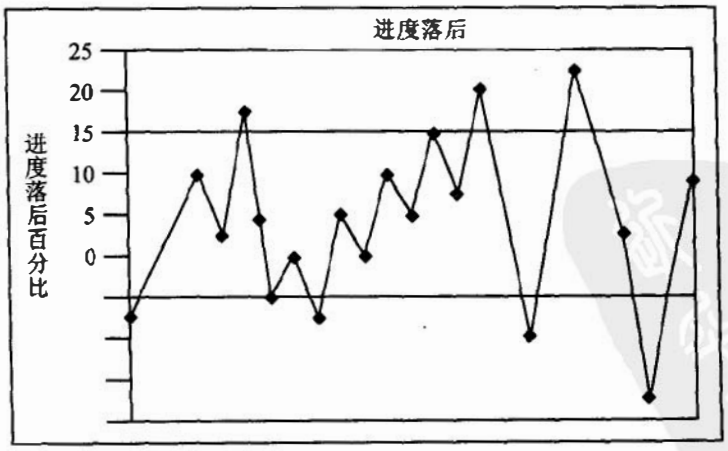


图 7.3　进度偏差的控制图

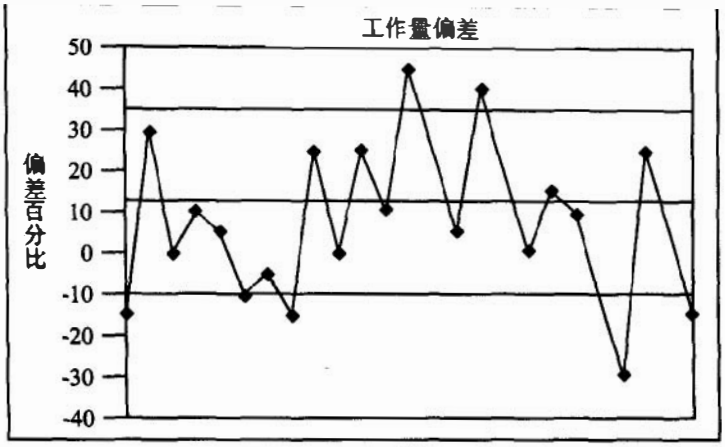


图 7.4 工作量偏差控制图

如果在里程碑处进度和工作量偏差超过了这些极限，可能隐含着项目遇到了麻烦，并且有可能不能满足其目标；在时间压力下，项目团队可能会采取不符合要求的捷径。在这种情况下，要求项目经理弄清楚引起偏差的原因，并根据需要采取纠正措施和预防措施。

这些控制极限都基于以往的数据和经验。你必须在制定计划时设定项目本身的极限，这些极限有可能高于组织的极限。

7.4 ACIC 项目的度量和跟踪计划

ACIC 项目对规模、工作量、故障和进度的标准指标都进行了度量。该计划对规模度量使用行计数器，对工作量度量使用 WAR 系统，对故障度量使用一个称为 BugsBunny 的故障控制系统，而对进度度量使用 MSP。

项目经理计划用 MSP 进行任务跟踪，并定期举行会议以监督各种任务的状态。问题分别被分类为联机问题、客户问题、业务经理问题和支持服务问题，并分别对它们进行跟踪。客户反馈信息（投诉以及表扬）都被记录下来。

每周向业务经理和客户发送状态报告。对于前 5 个里程碑处的偏差极限，工作量和进度设为 10%而故障设为 20%。对于其余里程碑处的偏差极限，工作量和进度设为 5%而故障设为 20%。里程碑报告也发给业务经理和客户。

跟踪和度量规划的最终结果包括在 ACIC 项目的项目管理计划中，有关内容请参见第 8 章。

7.5 小 结

在项目规划阶段，你必须决定打算如何监督项目的进展。进展监督对于保证项目朝着目标发展并在情况允许时采取纠正措施是必不可少的。项目监督通常要求执行某些度量。

下面是从 Infosys 公司的度量和跟踪规划中提取的一些经验教训：

- 制定规模度量、进度度量、工作量度量和故障度量的计划。这些指标对于大多数软件项目已经足够了。
- 将工作量进行分类，并用自动化系统按每类任务的代码收集工作量数据。为了避免因记忆问题引起的不正确性，必须及时记录工作量数据。
- 记录故障并跟踪它们，直到项目收尾。对于一个故障，还要记录其类型、检测阶段、引入阶段和严重性，以支持诸如故障排除效率、已交付项目的质量以及故障引入率等分析。
- 对于里程碑处的绩效分析，建立工作量、进度和故障的可接受的绩效变化极限。项目执行时，如果绩效超出这些极限，则必须保证管理干预。

虽然项目跟踪和度量是 CMM-2 的项目跟踪和监督 KPA、CMM-3 的综合项目管理 KPA 以及 CMM-4 的两个KPA所要求的，但是 CMM模型没有明确地表明需要规划这些度量。CMM 的基本原则规定必须对要执行的主要任务进行规划，因此这些KPA蕴涵了对度量进行规划。

7.6 参考文献

[1] S. D. Conte, H. E. Dunsmore, and V.Y. Shen. *Software Engineering Metrics and Models.* Benjamin/Cummings, 1986.

[2] S. H. Kan. *Metrics and Models in Software Quality Engineering.* Addison-Wesley, 1995.

[3] V. R. Basili and D. M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering.* 10(6), 1984.

[4] V. R. Basili, G. Caldiera, and H. D. Rombach. Goal question metric paradigm. In

Encyclopedia of Software Engineering, John J. Marciniak, editor. John Wiley and Sons, 1994.

[5] R. Grady and D. Caswell. *Software Metrics: Establishing a Company-wide Program.* Prentice Hall, 1987.

[6] Carnegie Mellon University/Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison-Wesley, 1995.

[7] N. Brown. Industrial-strength management strategies. *IEEE Software,* July, 1996.

[8] W. A. Florac and A. D. Carleton. *Measuring the Software Process: Statistical Process Control for Software Process Improvement.* Addison-Wesley, 1999.

[9] D. C. Motgomery. *Introduction to Statistical Quality Control, third edition.* John Wiley and Sons, 1996.

[10] D. J. Wheeler and D. S. Chambers. *Understanding Statistical Process Control,* second edition. SPS Press, 1992.

[11] W. Humphrey. *Managing the Software Process.* Addison-Wesley, 1989. ("SEI 软件工程丛书" 之《软件过程管理》，瓦茨·汉弗莱著，其英文影印版由清华大学出版社于 2002 年出版，中译版由清华大学出版社于 2003 年出版。)

[12] C. Jones. *Applied Software Measurement: Assuring Productivity and Quality,* second edition. McGraw Hill, 1996.

第 8 章　项目管理计划

项目管理计划（project management plan，简称 PMP）文档是项目经理承担的所有规划任务的核心。各种规划任务的结果都出现在该文档中，它是指导所有项目执行的基准文档。不应当将它与详细的项目进度计划混淆起来，后者仅仅表示任务的进度和分配。

用文档记录规划任务的结果，就可以对项目计划进行评审，找出其中存在的缺陷。在 Infosys 公司，项目通常由项目经理、SEPG 成员和高级管理人员组成的团体进行评审。在很多情况下，项目计划评审会揭示出显著的缺点，如果不纠正它们，就有可能给项目添加麻烦。对管理计划进行全面的评审是将潜在的问题扼杀在萌芽状态的最佳方法之一——对项目经理非常有用，尤其是那些没有多少经验的项目经理。

该文档还能在交流方面起到重要的作用。它向高级管理人员提供了项目目标和任务的总体视图，描述了为了实现这些目标应当如何对项目进行管理。它为项目团队提供了一个综合的项目视图及各团队成员的角色。

尽管我们已经讨论了大部分规划任务，但我们至尽还没有讨论团队与沟通（communication）问题，本章对它们进行了讨论。本章还讨论了 Infosys 公司用于记录计划的模板结构，研究了 ACIC 项目的项目管理计划。

8.1　团队管理

软件开发是一种团队工作。只有团队成员全力以赴并保持高度的积极性，并且整个团队稳定而高效的发挥作用，才能够实现高质量和高生产率。团队管理不仅仅是软件工程和项目管理问题，而且还涉及人员管理问题。人员管理是项目的一项重要内容，如果忽略它，会给项目和项目经理带来麻烦。

8.1.1　团队结构

在 Infosys 公司，通常采用层次性团队结构；一个团队由一个项目经理领导，

他向业务经理或者财务经理（或者同时向两者）报告。此外，一个典型的团队由开发人员（developer，简称 DV）、配置控制人员（configuration controller，简称 CC）和数据库管理人员组成；所有这些成员都向项目经理报告。一个大型项目可能还有模块领导，他们都向项目经理负责，并且他们底下还有一些开发人员。此外，还从现有的团队成员中选出一些人组成一个故障预防团队；这个团队负责执行与故障预防有关的任务。

前文已经讨论过，每个项目都有一个称为软件质量顾问（software quality adviser，简称 SQA）的 SEPG 成员与之联系。SQA 与项目经理（以及配置控制人员）广泛地沟通，但是不向项目经理报告。相反，SQA 有一个独立的报告渠道。

角色分配是一项复杂的任务。项目经理的目标是建立一个稳定的、自力更生的团队，有助于促进事业的发展，团队成员的才能发挥。因此，在确定团队结构时，项目经理必须考虑团队成员的个人需求和成长需求以及项目需求。下面给出了项目经理应该考虑到的一些有关人的因素：

- 团队成员的技能、工作背景和经验
- 团队成员的个人抱负和职位升迁
- 顾问和人员发展需要

初始团队结构以及每个团队成员的角色和责任都记录在 PMP 文档中。一个团队成员可以承担多重责任。项目执行期间，尤其是一个长期项目，可以根据各成员担当不同角色时的表现、个人抱负、积极性高低等对团队组织作相应的调整。

8.1.2 沟通

一个团队可能会为了一个共同的目标而在一起工作好几个月，因此团队必须凝成一团并且团队内部必须有良好的沟通机制。团队沟通从广义上可以分成两类：与项目相关的沟通和缓减压力的沟通（destressing communication）。一个出色的以人为本的项目经理会同时作好两种沟通的计划。

为了使团队成员及时获悉项目的进展和问题，一种方法是让他们访问项目状态报告及客户和业务经理的相关意见。除了这些正式报告外，根据团队的规模和项目的持续时间，Infosys 公司的项目经理用如下方法来增强团队沟通：

- 项目专用的公告牌：发布公告、通知、状态报告等等。
- 项目邮件列表

- 项目网站：发布文档、团队成员的主页、相关的技术论文和笔记以及自学培训材料
- 项目会议：关于简报和问题解决方案
- 团队成员关于他们的工作的最佳实践会议和讨论

此外，因为最终交付期限往往很短，并且每个人都有时间压力，所以压力会越来越大。旨在减压的沟通对于确保始终如一的积极性特别重要。很多项目经理都会计划一些任务，使这种“有趣的”沟通成为可能。例如，下面就是一些常用的方法：

- 项目晚会（组织预算支持所有团队）
- 生日宴会
- 诸如有奖竞赛和游戏等活动
- 非正式的、随心所欲的“面对面”交谈

8.1.3 团队建设

项目团队通常包含很多年轻人，团队和项目经理有责任加强这些团队成员的个人发展。帮助各团队成员成长对项目和组织也有好处。随着团队成员的技术和能力的提高，他们在项目后期的生产率会越来越高，并能承担更多的责任。当然，他们也变得更能胜任未来项目中的任务。为此，项目经理应采用如下这些方法：

- 岗位轮换
- 经验比较丰富的团队成员指导初来的成员
- 评审、评估和反馈
- 定期识别项目级的贡献
- 通过辅导、培训等帮助有困难的成员

8.2 客户沟通和问题解决方案

团队沟通旨在使团队及时掌握有关信息，保持团队的积极性。但是客户沟通怎样呢？客户是项目的发起人，因而也是一个主要的风险承担者。很多问题都是由客户和开发者之间的误解而引起的。开发团队和客户间的定期沟通有助于消除这些问题。

状态报告（第 11 章讨论）就是这样的一种沟通方式，旨在让客户清楚地了解正在进行的项目的状态。然而，这些报告无论多么详细，都是不够的。项目经理应当计划其他沟通方式，包括每周电话会议或者视频会议及定期发送电子邮件。在每周的虚拟会议上，项目领导与客户一起走查项目的状态报告，并说明项目的限制因素。讨论的重点是未决问题的解决方案。而客户寻求澄清真相，并在这些会议上阐明其观点。总之，除了发送报告外，通过定期交流，客户和开发团队就能在思想上保持一致。这样就防止了很多可能因误解而产生的问题。

尽管采用了定期交流的渠道，但是仍然会出现双方代表都不能解决的问题。这样的问题会潜在地延迟项目，因此必须提交它们。为了简化这种问题的解决方式，项目计划规定了客户方和 Infosys 方的提交方法。该计划还明确地表明了有关何时实施这些方法的策略。除了提供一种解决问题的机制外，这种提交方法和策略的规范也增加了双方的压力，使他们尽快地解决问题，并在需要时把它们提交给上级处理。

8.3 项目管理计划的结构

Infosys 公司提供的项目管理计划模板包括四个主要部分。项目提要（project summary）部分对项目进行了高度综述。它包含项目的开始和结束日期、项目领导、客户方联系方式、项目目标、向客户作出的关于里程碑和交付的软件的承诺，以及所作的假设等信息。所作的假设被显式地列出，因为它们通常充当某种风险源。同时还要描述付款的详情（以便业务经理可以跟踪它们）。无论从客户的角度还是从 Infosys 公司的角度，都要提及项目的目标，以便每个人都能明白为什么要执行该项目。

项目规划（project planning）部分列出执行各种项目规划程序的结果。它包含正在使用的开发过程、过程裁制备注、需求变更管理过程、需求跟踪能力计划、工作量和进度的估计及其基础，以及按技能、角色、每月工作的人员需求，或者这些需求的组合。它还规定了所需的开发环境、所用的工具以及任何项目专用的培训计划。本部分还给出了质量计划和风险管理计划。

项目跟踪（project tracking）部分定义要采用的度量措施以及用于记录数据的系统、要采取的各种项目跟踪任务、进展报告的频度和性质以及提交程序。

项目团队（project team）部分定义项目团队及其结构，以及各成员的角色和责任。

8.4　ACIC 项目计划

本节介绍案例研究 ACIC 项目的项目计划。PMP 的第 1 部分给出主要参与者、里程碑和一个项目综述。（第 4 章说明了如何确定进度和里程碑。）

第 2 部分包含关于规划的详情。首先，它列出了过程规划的结果。计划表明将要使用的“开发过程”以及为项目所做的裁制工作。这个项目将使用 Rational Rose 统一过程（Rational Rose Unified Process，简称 RUP），因为这是客户的要求。RUP 包含四个主要阶段：初始阶段、细化阶段、构造阶段和移交阶段。细化和构造阶段通常迭代式地完成。细化阶段的主要任务是分析与设计，而构造阶段的主要任务是编码和测试。项目将进行 2 次细化迭代和 3 次构造迭代。为了满足这种方法，需要对 Infosys 公司采用的标准开发过程进行裁制。裁制备注规定了对标准过程进行的裁制工作。

对于需求变更管理，计划规定将变更申请记录在哪里以及处理它们的过程。它还规定，如果任何一种变更申请需要总工作量的 2%以上，则需要对它重新进行估计。对于需求可跟踪性，它表明将使用一种工具。

工作量估计子部分给出了工作量估计及估计的基础。（第 4 章讨论了 ACIC 项目的估计。）它与工作量估计一起，规定了项目的进度（这与客户要求的里程碑相同）以及满足进度要求所需的人员。人员需求还根据技能进行规定。还给出了团队的培训计划，以及延迟某人进行培训的条件。

质量计划子部分给出了质量目标和故障的中间目标。因为项目本身设定了更高的目标，所以同时也给出实现那些目标的策略，以及项目准备执行的评审。

规划部分也规定了所需的硬件资源和软件资源以及风险管理计划（参见第 6 章）。

项目跟踪部分（第 3 部分）提到将用 MSP（MicrosoftProject）进行任务调度、工作分配和状态监督。BugsBunny 是一个内部开发的工具，将用来跟踪项目所属的问题。项目经理负责评审这些问题并确保解决它们，但是不涉及与业务经理相关的问题。BugsBunny 还将用来跟踪客户投诉和反馈信息。状态报告每 2 周发送一次，并且客户将在每个里程碑处接到 1 个报告。在里程碑处，计划进行详细分析。在双方都建立了问题的提交渠道。

第 4 部分规定团队结构。团队由项目领导（Project Leader，简称 PL）负责（本

书别的地方称为项目经理），他管理一个故障预防（defect prevention，简称DP）团队、一个模块领导及配置控制人员。DP团队负责分析故障并建议预防它们的方案。（第5章讨论了DP规划，第11章说明了DP任务的执行。CC的作用将在第9章作进一步的讨论。）团队结构规定：软件质量顾问与团队有关，但他不向PL报告。该计划还规定团队中每个人员的角色和责任。

ACIC项目的项目管理计划

1. 项目提要

1.1 项目综述

ACIC是一个基于美国的投资公司。该应用包括两个组成部分：一个是ACIC网站上的经纪账户开户（Brokerage Account Opening）应用，它允许任何一个Internet用户开启一个ACIC经纪账户；另一个是账户开户和维护应用，主要是让ACIC代表对以书面格式申请的账户进行开户。这是一个Internet应用。该应用将提供诸如账户历史查看和账户结余、状态和任务信息。这将允许ACIC有效地发展为一个客户/账户服务应用，而不仅仅是一个账户开户引擎。这是对一个现有应用的增强；该应用的早期开发也是由Infosys完成。

项目代码	项目名称	客户
xxxxxxxx	ACIC项目	ACIC公司

项目领导（PL）	配置控制人员（CC）	业务经理（BM）	候补PL	候补CC
BB	SB	HR	BJ	HP

项目类型	平台	阶段数
开发项目	Java、Win NT、DB2	4个阶段

项目开始状态（包括联机、脱机）		项目收尾日期	估计的总收入
联机	脱机		
2000年4月3日	2000年5月15日	2000年10月3日	xxx xxx美元

项目和客户联系人员

名字和职称	电话号码	传真号码	E-mail

项目范围

- 提供一种高效而快速的账户维护方法
- 允许代表访问信息
- 向客户代表提供账户状态、价值、定单状态和交易任务的完整视图
- 增加更新过程的智能性
- 提供一个能够显示所需账户的历史要素的接口
- 提供关闭和重新激活一个账户的能力

项目对客户的增值

- 该项目将允许 ACIC 有效地发展成一个客户账户服务应用，而不仅仅是一个账户开户引擎

Infosys 目标

- 通过按时交付高质量的软件，增强与 ACIC 公司的关系
- 通过发展有关 ACIC 产品和系统的专门知识，成为首选的软件商

1.2　向客户作出的承诺

序号	里程碑日期	里程碑	提交的结果
1	2000 年 5 月 26 日	项目开始：需求分析结束	业务分析和需求规范、用例目录、屏幕、迭代计划
2	2000 年 5 月 15 日-6 月 23 日	细化阶段：第 1 次迭代	顺序图、类图、源代码、下一个周期的计划
3	2000 年 6 月 26 日-7 月 7 日	细化阶段：第 2 次迭代	补充规范、顺序图、类图、体系结构文档、源代码、下一个周期的迭代计划
4	2000 年 7 月 10 日-7 月 21 日	构造阶段：第 1 次迭代	源代码、评审报告、测试报告、下一个周期的迭代计划
5	2000 年 7 月 20 日-28 日	构造阶段：第 2 次迭代	源代码、评审报告、测试报告、下一个周期的迭代计划
6	2000 年 7 月 31 日-8 月 8 日	构造阶段：第 3 次迭代	源代码、评审报告、测试报

其他要求

序号	要求
1	该项目将遵循 Rational 统一方法（RUP）

1.3 假设

规划时作出的假设

- 迁移到 VisualAge for Java 3.0 不由该团队完成
- 业务合伙人的智能更新仅与应用的维护部分结合，而不与“账户开户”引擎结合
- 有资格的人员将批准用 Rational 统一过程方法实现该项目
- 在项目生命期中，功能和技术需求的变更可能会对进度产生影响。由于这些变更而导致对成本或者进度的任何影响，都将通知 ACIC
- ACIC 评审人员将用 7 天时间来改进一个里程碑文档。如果在这期间没有收到注解，则认为它已经得到批准

2. 项目规划

2.1 项目过程

项目遵循的标准过程

该项目将遵循 Infosys 的标准开发过程。然而，我们将用 Rational 统一过程方法（RUP）对这一过程进行增强，因为这是客户的要求。为了符合RUP，将对标准的开发过程进行裁制。

裁制备注

与标准过程的偏差	增加/修改/删除	偏差的原因
只有那些在一次特定的迭代中将被细化的用例才在那次迭代中被采用	修改	基于迭代的开发
在前几次迭代中，将增量地开发逻辑对象模型	修改	遵循RUP方法
在前几次迭代中，将增量地开发物理对象模型	修改	遵循RUP方法
物理数据库设计可以在后面几次迭代中精化	修改	遵循 RUP方法
在每次迭代中都要开发单元测试计划	修改	正在使用迭代方法
故障按每次迭代进行记录	修改	正在使用迭代方法
需求跟踪将根据 Requisite Pro 工具完成	修改	遵循RUP方法
没有图象文档和业务用例，因为我们以范围文档开始，这所起的作用是相同的	修改	背离 RUP

需求变更管理过程

变更申请跟踪

- 客户申请的变更将记录在 ChangeRequest.xls 文档中，并分析它们对项目的影响。一个变更申请书将提交给客户以得到他的确认。经客户确认的变更申请将添加到项目合同中作为其附录
- 主要变更通常对项目的进度/按时交货有影响。客户需要正式地批准这些变更
- 因为这是一个短期项目，如果任何一个变更申请或者一组变更申请超过了项目估计的工作量的 2%，则必须重新评估项目进度和工作量

需求跟踪性

使用 Requisite Pro 工具。

2.2 项目估计的规模和工作量

估计标准

程序/函数（用例）	标准
简单用例	3 个或 3 个以下事务
中等复杂用例	4-7 个事务
复杂用例	>7个事务

用例编号	说明	复杂性	用例编号	说明	复杂性
1	屏幕导航	复杂	14	更新一个账户的细节	中等复杂
2	更新个人详情	中等复杂	15	维护一个账户的任务	复杂
3	增加地址	中等复杂	16	维护一个账户的备忘录	简单
4	更新地址	复杂	17	查看团体情况的历史	复杂
5	删除地址	复杂	18	查看账户情况的历史	复杂
6	添加电话号码	中等复杂	19	查看选项等级和服务选项的历史	简单
7	更新电话号码	复杂	20	查看任务和备忘录的历史	简单
8	删除电话号码	复杂	21	查看角色的历史	复杂
9	添加 E-mail	中等复杂	22	查看账户细节	简单
10	更新 E-mail	中等复杂	23	查看账户的所有权	复杂
11	删除 E-mail	中等复杂	24	查看一个账户的紧迫定单	复杂
12	更新一个团体的就业情况	中等复杂	25	关闭/重新激活账户	简单
13	更新一个团体的财政情况	中等复杂	26	智能更新 ACIC 的商业伙伴	复杂

项目估计的构建工作量

程序/函数	工作量（基于以往项目的数据）	单元数	总的构建工作量（人日）
简单用例	1人日	5	5
中等复杂用例	5人日	9	45
复杂用例	8人日	12	96
总工作量			146

按阶段的工作量估计

任务/阶段	人日	占总工作量的百分比（%）
需求	50	10
设计	60	12
构建	146	29
集成测试	35	7
回归测试	10	2
验收测试	30	6
项目管理	75	15
配置管理	16	3
培训	50	10
其他	40	6
估计的工作量	501	100

按迭代过程的工作量估计	人日	占总工作量的百分比（%）
项目开始	25	5
初始阶段	24	5
细化阶段：第1次迭代	45	9
细化阶段：第2次迭代	34	7
构造阶段：第1次迭代	27	5
构造阶段：第2次迭代	24	5
构造阶段：第3次迭代	21	4
移交阶段	110	22
项目收尾	10	2
项目管理	75	15
配置管理	16	3
培训	50	10
其他	40	8
估计的总工作量	501	100

2.3　进度计划

在对客户的承诺部分，规定为里程碑。

2.4　人员

按角色人员分配

角色	所需人数	日期
PL	1	2000 年 5 月 4 日
联机协调者	1（50%时间）	2000 年 5 月 4 日
模块领导	1	2000 年 5 月 15 日
开发人员	3	2000 年 5 月 15 日
开发人员	1	2000 年 7 月 17 日
开发人员	1	2000 年 8 月 1 日
开发人员	1	2000 年 8 月 14 日
总人数	9（实际为 8.5）	

按技能和经验人员分配

领域	总人数	0-12 个月经验	>12 个月的经验
Java	7	7	0
DB2	2	0	2
总人数	9	7	2

人员需求计划

月份	脱机开发	联机部署	总人数
2000 年 5 月	4	1（50%）	5
2000 年 6 月	5	1	6
2000 年 7 月	5	1	6
2000 年 8 月	8	1	9
2000 年 9 月	7	2	9
2000 年 10 月	3	2	5

2.5　开发环境

硬件	软件
NT Server	WinNT
大型机	DB2
Intel PC	VisuageAge for Java，Java，Win NT

2.6 所需的硬件和软件资源

项目描述	所需数量	日期
PC（128 RAM）	6	2000 年 5 月 1 日
服务器上有 1GB空间	1	2000 年 5 月 1 日
VisuageAge for Java	6	2000 年 5 月 4 日
DB2	6	2000 年 5 月 4 日
Rational Rose	5	2000 年 5 月 15 日
Requisite Pro	1	2000 年 5 月 15 日

2.7 工具

工具列表	
项目要开发的工具	无
项目使用的内部工具	BugsBunny，WAR

2.8 培训计划

培训领域	持续时间	免培训条件
技术领域		
Java 语言	7 天	如果已经培训过
VisualAge for Java	3 天	作为初始培训的一部分
Java Applets	4 小时	如果已经培训过
Java Swing	4 小时	如果已经培训过
Persistence Builder	4 小时	如果已经培训过
Rational Rose 和 Requisite Pro	8 小时	必须培训
OOAD	1 天	如果已经培训过
业务领域		
系统评估	7 天	如果已经培训过
与过程相关的领域		
质量系统	3 小时	如果已经培训过
配置管理	2 小时	如果已经接受过CC 培训。其他进行在职培训
小组评审	4 小时	如果已经培训过。
故障预防	4.5 小时	必须培训
SPC 工具	4.5 小时	如果已经培训过
RUP 方法	2 小时	必须培训

2.9 质量计划

质量目标

项目质量目标

目标	值	设置目标的基础	组织范围的标准
引入的总故障数	145	0.033 个故障/人时。这要比 Synergy 好 10%，后者为 0.036 个故障/人时	0.052 个故障/人时
质量（验收故障密度）	5	估计的总故障数的 3%或者以下	估计的总故障数的6%
生产率	57	生产率比 Synergy 提高 3.4%	50
进度计划	按时交货		10%
质量成本	32%	31.5%	32%

估计检测到的故障数

评审/测试阶段	估计检测到的故障数	占估计的总故障数的百分比(%)	估计基础
需求和设计评审	29	20%	参考同类项目的估计（Synergy）和 PCB
代码评审	29	20%	参考同类项目的估计（Synergy）和 PCB
单元测试	57	40%	参考同类项目的估计（Synergy）和 PCB
集成和回归测试	25	17%	参考同类项目的估计（Synergy）和 PCB
验收测试	5	3%	参考同类项目的估计（Synergy）和 PCB
估计检测到总故障数	143	100%	

实现质量目标的策略

策略	期望的效益
用标准的故障预防指南和过程进行故障预防；采用 Synergy 开发的编码标准	故障引入率减少 10-20%，生产率提高2%
小组评审前面几个或者逻辑上复杂的用例 项目领导、开发人员和一个顾问共同评审设计文档或者开发的第一个程序	质量提高，因为总故障排除效率将提高；生产率会有所提高，因为故障将在早期被发现
引入 RUP 方法，并迭代式地实现项目。在每次迭代后，执行里程碑分析和故障预防任务	故障引入率大约减少了 5%，并且总生产率提高 1%

评审

评审时机	评审条目	评审类型
项目规划结束时	项目计划 故障控制系统(DCS)计划 项目进度计划	小组评审 软件质量顾问（SQA）评审 软件质量顾问（SQA）评审
项目规划结束时	CM计划	小组评审
完成90%的需求分析时(这必须在细化阶段的第 1 次迭代结束时)	业务分析和需求规范文档、用例目录	小组评审
完成 90%的设计时(这必须在细化阶段的第 2 次迭代结束时)	设计文档、对象模型	小组评审
每次迭代开始时	迭代计划	个人评审
详细设计结束时	复杂的程序规范和第一次产生的程序规范，包括测试案例、交互图	小组评审
前面几个程序编码以后	代码	小组评审
自我测试一个过程以后	代码	个人评审
单元测试计划结束时	单元测试计划	个人评审
集成测试开始时	集成测试计划	小组评审

2.10 风险管理计划

表 6.4 ACIC 项目的风险管理计划

序号	风险	概率	影响	风险暴露度	风险缓和计划
1	需要客户的数据库设计师和数据库管理员的支持	0.5	8	4	仔细地规划每组任务所需的时间，并事先给予充分的注意 联机协调者与这些小组紧密合作
2	因为第一次使用 RUP，团队对 RUP 的理解可能不全面	0.9	3	2.7	与 Infosys R&D 实验室专家密切合作 在整个项目中，保持与客户沟通并提交任何进度或者工作量偏差 对团队进行RUP方法培训
3	人员流失：团队成员可能临时离开	0.3	7	2.1	分配任务时，使多个人从事项目中的每个单元和用例
4	通过链路与客户的大型机 DB2通信：链路可能不如期望的那样有效	0.1	8	0.8	执行额外的代码评审、桌面检查等以使对链路的依赖最小 链路绩效下降时立即提交

3. 项目跟踪

3.1 度量计划

指标	度量单位	所用的工具
规模	LOC、FP、S/M/C 计数	行计数器
工作量	人日	WAR
故障	故障数	BubsBunny
进度	经过的时间	MSP

3.2 任务跟踪

任务	程序
任务调度	PL 用 MS Project 调度任务 需要将进行精化和重新调度
任务分配	使团队成员知道最新的进度计划。进度计划一旦上载到 WAR-MSP 系统，任务将在各自的 WAR 中出现
任务状态跟踪	每天执行任务跟踪
项目会议	每周一次
因果分析会议	每次迭代后

3.3 问题跟踪

问题类型	在哪里记录	谁记录	谁评审，何时评审	何时提交
联机问题	IssueTracker.xls	任何一个项目成员	PL，每天评审	2 天内
客户问题	IssueLog.xls	联机团队，PL	PL，每天评审	2 天内
业务经理问题	每周状态报告	BM	BM，PL，每周评审	5 天内
支持服务问题	申请跟踪器	任何团队成员	支持服务，每天评审	2 天内

3.4 客户反馈信息

项目	记录和跟踪过程
客户反馈	AM/PL 处理客户反馈信息。BM保存它
客户投诉	接到的客户投诉将输入到 CustomerComplaints.xls 中，并用它进行跟踪

3.5 质量跟踪

质量任务	措施
故障跟踪	用 DCS 记录故障，并用它跟踪故障，直到结束
评审（需求、概要设计、详细设计）	根据质量计划中的项目目标进行检查
代码评审	通过 SPC 工具根据每个程序的极限进行检查
独立的单元测试	通过SPC工具根据每个程序的极限进行检查
集成测试/系统测试	根据质量计划中的项目目标进行检查

3.6 高级管理人员（BM）评审

序号	评审项目	评审频度
1	进度计划	每次版本变化时
2	项目计划	作出重大改变时
3	里程碑报告	里程碑结束时

3.7 状态报告

向谁报告	报告频度
业务经理	每周周一通过电子邮件报告
客户	每周周一

3.8 里程碑处的偏差极限

实际与估计间的偏差	前面5个里程碑	其余里程碑
工作量	10%	5%
进度	10%	5%
故障数	20%	20%

3.9 向客户报告

- 里程碑报告和每周状态报告
- 需要澄清的问题
- 如果有任何需要提交的问题，则提交它们

3.10 向 BM 报告

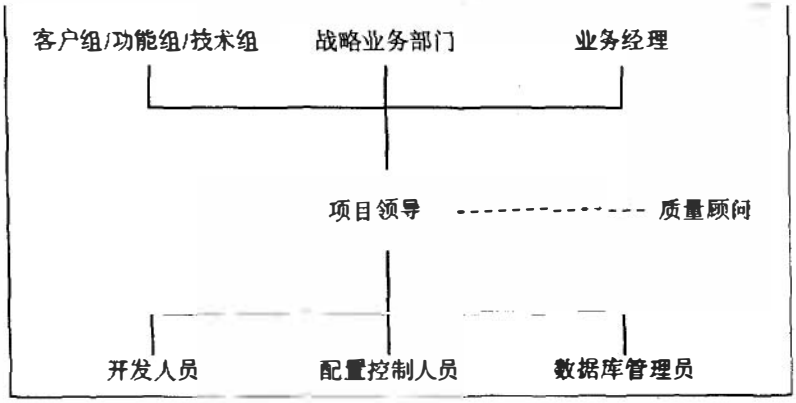
- 客户反馈
- 里程碑和每周状态报告
- 需要澄清/注意的问题
- 如果有任何需要提交的问题，则提交它们
- 需求变更量及其估计的工作量
- 计划的主要变更

3.11 提交程序

提交地点	提交期限	提交人姓名	提交人的职称
在 ACIC	3天内	Xxxx	项目经理
在 Infosys	3天内	Xxxx	财务经理
在 Infosys	3天内	Xxxx	业务经理

4. 项目团队

4.1 项目组织



4.2 项目团队

序号	姓名缩写	责任	开始日期	期望的结束日期
1	BB	项目经理	2000年4月4日	2000年11月3日
2	KP	联机协调者	2000年4月4日	2000年11月3日
3	BJ	模块领导、侯选项目领导	2000年5月15日	2000年11月3日
4	SP	配置控制人员	2000年5月22日	2000年10月13日
5	DD	开发人员	2000年5月22日	2000年9月29日
6	HP	开发人员，侯选配置控制人员	2000年5月22日	2000年9月29日
7	NA	开发人员	2000年7月17日	2000年11月3日
8	SH	开发人员	2000年8月1日	2000年9月15日
9	AL	开发人员	2000年8月14日	2000年8月31日
10	JP	开发人员	2000年12月1日	2000年9月22日
11	SDS	财务经理	2000年4月4日	2000年11月3日
12	SB	SQA	2000年5月15日	2000年11月3日

4.3 角色和责任

角色	责任
业务经理（BM）	● 解决提交的问题 ● 检查项目状态 ● 参与关键技术评审
客户	● 评审设计 ● 解决提交的问题 ● 验收测试计划和测试
财务经理（AM）	● 客户感到满意 ● 业务增长 ● 项目的财务计划 ● 销售和推销 ● 与培训相关的问题 ● 与雇员相关的问题
项目经理（PM）	● 项目规划和调度 ● 设计 ● 客户交互 ● 评审 ● 测试 ● 报告 ● 任务分配和跟踪 ● 与 SEPG 的软件质量顾问交互 ● 保证按合同交付产品 ● 根据需要，与其他部门的交流 ● 保证正确地解决未解决的问题/客户投诉 ● 保证项目成员得到充分的培训
模块领导（ML）	● 设计 ● 开发 ● 测试 ● 报告
故障预防（DP）团队	● 引起团队对故障及其预防的注意 ● 分析故障数据 ● 确定减少故障引入的方法
开发人员（DV）	● 用例的详细设计 ● 开发 ● 单元测试和集成测试
配置控制人员（CC）	● 准备 CM 计划 ● 根据 CM 计划管理配置
SEPG 的软件质量顾问（SQA）	● 过程顾问 ● 质量保证（审计） ● 安装度量工具和培训项目人员 ● 根据需要参与项目计划的过程的评审
联机协调者	● 解决来自客户/脱机开发的任何问题 ● 开发期间的支持

5. 参考文献

略。

6. 缩略语

略。

8.5 小 结

重要的是项目经理规划好团队和沟通管理，并准备一份最终的综合性项目管理计划文档。

下面是本章讨论的重要问题：

- 软件项目是一项团队工作。管理团队时，必须考虑到项目的目标以及团队成员的目标。
- 定期举行团队内交流以及与客户的交流。明确地规划双方的提交渠道，以解决原本很难解决的问题。
- 将各种规划任务的结果记录在一个综合性计划中。对此文档进行评审，以识别出潜在的规划缺点，并让高级管理人员根据其经验对它进行改进。
- 用文档记录计划，它必须包含项目的信息、将要采用的过程、变更管理方法、工作量估计、里程碑、风险管理计划、质量计划、项目跟踪和报告计划、项目以外的问题解决的提交渠道，以及团队组织等。

关于CMM，一个用文档记录的项目管理计划是CMM-2的项目规划及项目跟踪和监督KPA的一个需求。本章讨论的方法和项目管理计划的内容也满足CMM-3的集成软件管理和组间协调KPA的需求、CMM-4的两个KPA的需求以及CMM-3的培训KPA的某些特征。

第9章 配置管理

在一个软件项目中，由于工作产品的发展和需求变更而产生的变化，会接连不断地发生。所有这些变化最终将体现在包含源代码、数据或者文档的文件中所发生的变化。在一个软件项目的大量文件由多人共同创建的情况下，如果他们都可以对这些文件进行变更，则必须对这些变更进行正确地控制，否则问题会复杂化。考虑下面这些取自各种不同项目的情况：

- 假定客户提出了两种不同的变更申请。项目经理将一种申请交给 Rao 去实现，而将另一种申请交给 Meera 去实现。两人都必须修改模块 X 的代码。当 Meera 完成她的修改时，她保存了模块 X 的文件，并不经意地地覆盖了 Rao 几天前完成的修改。
- 一个模块由三个团队成员开发代码，其最后交付期限是周五。他们计划在最后 2 天集成经过单元测试的代码。Subbu 是几个关键函数的开发者，他突然在周二晚上离开团队要去办一件紧急事情。次日，模块领导和团队成员花了很长时间来研究 Subbu 的文件。他们设法标识出那些包含 Subbu 正在开发的各种函数的文件，但是他们发现这些文件有很多个版本。因此，他们不得不让其中一个团队成员来解决该问题，该团队成员一直到周末才完成该任务。他以 Subbu 开发的某个版本的程序开始，开发了 Subbu 的程序单元并对该单元进行了测试，他重做了 Subbu 离开团队前几乎已经完成的工作。最后，模块的交付时间推迟了 3 天。
- Srinath 团队的精神状态很好。他们按时完成了项目开发，并且最后的测试也没有出现问题。软件被交给客户去实施。次日，Srinath 收到了来自用户和客户的气愤的电子邮件，他们报告了软件存在的问题。经过团队紧张的工作，终于找到了原因：软件的发行版中的某个关键组件使用了旧版本。

诸如此类的问题可以在许多组织中发现。配置管理（configuration management，简称 CM）也称为软件配置管理（software configuration management，简称 SCM），是项目管理的一项内容，主要涉及对变更进行系统地控制以防止此类问题的发生。

本章首先讨论一些与 CM 有关的基本概念，然后讨论 Infosys 公司遵循的CM

过程。此外，还给出了案例研究的CM计划。

9.1 配置管理概念

CM对于实现项目中的一个基本目标很重要：将高质量的软件产品交给客户。这种交给客户的“软件”是什么呢？它至少包含构成源和对象代码的源和对象文件、从这些文件建立工作系统的脚本及其相关文档。在项目执行期间，文件变更会导致不同版本的软件的建立。在这种情况下，程序管理员如何保证组合了正确版本的源代码文件，而又没有遗漏源代码呢？又如何保证发送了与最终的源代码一致的正确版本的文档呢？所有这一切都通过正确的CM得到保证。

CM的一个主要目标是对软件系统不断变化的配置进行管理。[1,2]在一个项目中，一个程序的发展要经过很多阶段。最初程序员开发它时，程序处于开发状态（或者“私有”状态）。在程序员对程序感到满意之后，程序就进入了“准备单元测试”状态。只有当程序达到这个状态时，才能对它进行单元测试。在程序经过单元测试之后，程序员必须修复发现的任何一个问题。然而，如果成功地通过了单元测试，则程序进入“准备系统测试”阶段。只有在所有程序都达到这个状态时，才可以开始系统测试。此外，如果在系统测试阶段发现故障，则程序状态返回到“私有”状态；否则程序进入“准备验收测试”状态。如果验收测试成功，则所有程序的状态改变成“准备发行”状态，表明可以发行它们，并进入“生产使用”状态。一旦程序被发行并处于生产使用状态，所有程序（及相关的文档）都进入“基准（baselined）”状态，它代表生产系统的状态。

除了正常的软件开发期间发生的变更，需求变更申请还可以通过提交渠道进行提交，并且它们的实施可能会改变程序。当一个项目有大量可以被变更的条目时，可能会要求开发人员采取很多措施；只有在CM过程的正确支持下，才能执行这些措施。

为了更好地理解CM，让我们考虑一下项目要求的各种CM功能。尽管这些需求依赖于项目的性质和具体情况，但是某些基本功能是可以确定的。下面列出了其中一些功能，以及那些可能需要它们的环境。这些功能比Humphrey定义的CM功能更加具体[3]。

- 给出程序的状态。你需要此信息来决定何时开始测试或者何时发行软件。
- 给出一个程序的最新版本。假设必须对一个程序进行修改。显然，必须

在最后一个版本中实现修改；否则，前面的修改可能会被丢失。

- 处理并发更新申请。在响应两个不同的变更申请时，两个程序员可能会并发地修改同一个程序。其中一个的修改有可能覆盖另一个的修改。为了避免这种情况的发生，需要有访问控制机制，使得在某一时刻只有一个人能够对程序作出修改。如果允许多个人并行地修改，则应当实施协调过程以保证最后的版本体现出所有的修改。
- 取消一个程序变更。对一个程序作出某种变更后（为了实现某个变更申请），但后来出现某个申请，需要撤消这种变更申请。
- 防止未授权的变更或者删除。程序员可以决定变更某些程序，但是发现这种变更有负面影响。为了拒绝接受未授权的变更，需要有访问控制机制。
- 提供需求变更申请和程序变更之间的可跟踪性。假设一个需求变更申请规定要对三个程序进行修改，并且这些修改任务已经分配给三个团队成员。项目经理如何保证他们正确地实现了该变更申请呢——即，如何保证所有程序都已被修改过，并且修改过的程序正处于生命期的“准备发行”状态呢？该问题的答案是要求一种跟踪变更申请的机制，它可以规定所有要进行修改的程序以及每个程序的状态。
- 取消一个需求变更。某个已经被实现的需求变更申请（通过改变很多程序），事后可能需要被撤消（也许因为客户不喜欢变更后的新特征）。
- 显示相关的变更。假设在某个程序中发现了一个 bug，并且怀疑这个 bug 是由于实现某个变更申请而引起的。则必须检查所有因那个变更申请而导致的变更。
- 收集当前系统的所有源代码、文档和其他信息，以便在文件遭到破坏或者系统崩溃时能够恢复所有文件。同样，可能需要对现有系统（正在运行的系统）作出某种变更，所以必须收集所有代表当前系统的源文件和文档。

上面所列的情况是一个项目中经常发生的需要 CM 过程支持的情况。此外，如果某个软件产品有多个版本同时存在，并且每个版本的软件产品使用一个不同版本的程序，则可能出现其他与变更相关的情况，要求 CM 具备额外的功能（例如，处理变体）。尽管如此，但 CM 的主要目标是提供处理前面列出的各种情况的机制。这些机制包括：

- 文件命名和组织的约定
- 版本控制

- 变更申请的可跟踪性
- 访问控制
- 协调过程
- 修改登记程序

根据标准约定命名程序文件（和文档文件），并把文件保存到指定的目录中，这有助于迅速发现某个所需的文件。正确的命名（例如，通过使用标准的扩展名）还有助于开发人员容易地理解文件内容的本质，而不必查看文件内容。此外，将程序根据它们的状态分别保存到不同的目录中，这有助于开发人员轻松地识别出程序的状态。

版本控制是CM的一个关键问题[1,2,4]，有许多工具可以帮助管理这个任务。版本控制有助于保护程序的旧版本，而不管程序如何变化。如果没有这么一种机制，则系统不能支持许多必须的CM功能。

变更申请跟踪性机制提供了从一种需求变更申请到随后的程序变更的映射，有助于需求变更的管理。为了从一个更高找到变更申请之前的变更，一种有效的办法是借助于修改日志。

访问控制机制保证只有被授权的人员才能修改某些文件，并且在某一给定时间只有一个人能够对一个文件进行修改。协调程序规定了如何合并对一个程序独立进行的两次修改，建立一个新版本，以体现这两次修改。

如果这些机制都被提供了，则可以令人满意地处理前面给定的情况。这些情况中的一些需要采用多种机制。例如，取消一个需求变更牵涉到一种表示从需求变更到随后的程序变更的可跟踪性机制，以及一种版本控制机制以切实地取消变更。

一些CM机制可以得到某个工具的支持，而另一些可能需要用户显式地执行它们。例如，版本控制可以通过一个工具而实现，但是获取一个程序的状态可能要求程序员显式地维护这种信息。CM过程定义了实现这种机制的所有步骤，并说明了如何在项目中使用这些机制。

前面所讨论的内容主要集中于程序。项目所产生的文档（诸如需求文档、设计文档和计划）也需要配置管理。在一个项目的正常执行期间，一个文档往往会经历3个状态："正在开发"、"正在评审"和"基准状态"。状态转移很明确，并且类似于Whiggift建议的一种方法。[4]CM过程还必须实现文档的状态图。

9.2 配置管理过程

CM 过程定义了支持 CM 机制必须执行的任务序列。与项目管理中的大多数任务一样，Infosys 公司采用的 CM 过程的第一阶段也是规划，确定那些需要进行配置管理（CM）的条目（称为配置项）、保存它们的位置、变更控制的程序等等。项目经理或者配置控制人员（CC）准备这个计划。然后该过程必须被执行，也许通过部署一个工具（工具的使用也要加以规划）来实施。最后，因为任何一个 CM 计划要求项目人员维护版本、将配置项保存到合适的位置，以及正确地作出变更等方面的训练，所以监督配置项的状态和执行 CM 审计是CM 过程的其他任务。本章讨论 Infosys 公司采用的CM 过程的这三种任务。

9.2.1 配置管理规划和制定

配置管理规划涉及确定配置项和规定用来控制和实现其变更的程序。确定配置项是任何一种 CM 的基本任务。[2,3,4]典型的配置项如需求规范、设计文档、源代码、测试计划、测试脚本、测试程序、测试数据、项目使用的标准（诸如编码标准和设计标准）、验收计划、诸如 CM 计划和项目计划等文档、诸如用户手册等用户文档、以及诸如培训材料等文档、合同文件（包括支持工具，诸如编译器或者内部工具）、质量记录（评审记录、测试记录）和 CM 记录（版本记录、状态跟踪记录）。客户提供的任何产品或者购买的任何东西，只要它将成为发布的软件的一部分（称为“被包含的软件产品”），那么它也是配置项。

规划期间，CM 的配置项类型已经被识别出，但是没有准备好这些配置项的详细列表。这种疏忽反映了这样一个事实，即在 CM 规划期间某些配置项可能不为人知（这出现在项目开始时）。为了使配置项的正确命名更加容易，CM 规划阶段制定了 CM配置项的命名约定。除了命名标准外，项目经理还必须计划版本编号。当一个配置项发生变化时，不是用新项替代旧项，而是旧项被保存起来，并建立一个新项。这种方法导致一个配置项存在多个版本，因此需要有版本号分配策略。如果使用了一种 CM 工具，则有时可以用它处理版本编码问题。否则，必须在项目中显式地处理编码问题。

规划期间，项目经理必须确定如何维护一个程序的状态。一种收集不同状态的配置项的方法是为它们建立不同的目录。某个状态的所有配置项都保存在那个状态的目录中。当一个程序的状态发生变化时，则将那个程序从旧状态的目录移到新状态的目录。这是一种基本方法，不要求使用任何工具来维护状态信息。然

而，如果使用一个 CM 工具，则管理程序状态的目录结构依赖于工具。在规划阶段，项目经理必须设置用于管理程序状态的目录结构，如果 CM 工具有一些需求，则必须牢记这些需求。

配置控制人员或者项目经理进行 CM 规划。只有在项目已经开始，并且已经用文档明确地记录了运行环境和需求规格说明，CM 规划才可以开始进行。该阶段包括如下一些任务：

- 标识配置项，包括客户提供的配置项和购买的配置项。
- 定义配置项的命名机制和编码机制。
- 定义 CM 所需的目录结构。
- 定义访问限制。
- 定义变更控制过程。
- 标识并定义 CC 或者配置控制委员会（Configuration Control B ard，简称 CCB）的责任和权限。
- 定义一种配置项状态的跟踪方法。
- 定义一种备用过程。
- 如果需要，定义一种协调程序。
- 定义一个发行程序。
- 定义一个存档程序。
- 确定配置项将被移入基准库的时机。

这一阶段的结果是 CM 计划。CC 负责 CM 计划的实施。根据正在开发的系统的规模，CC 的角色既可以是一份兼职工作，也可以是一份全职工作。CC 也可能负责版本管理、版本归档、在需要时找到并发行合适的版本，等等。

在某些情况下——当团队规模很大时，或者两个或者多个团队或者小组参与软件或者接口系统的相同部分或者不同部分的开发时——这时可能有必要有一个 CCB。这个委员会包括来自每个团队的代表。一个 CCB（或者一个 CC）是 CM 的基础[3]，并且 CM 计划必须明确地定义CC或者 CCB 的角色或者责任。这些任务还取决于文件系统的类型和所用的工具。

前文已经提到过，CM 要求限制对某些状态下的某些项目的访问。例如，必须限制程序员访问和修改基准库中程序的权力。因此，规划任务必须规定 CC、项目经理和开发人员的访问权力。

规划期间还要建立变更控制的策略和程序。这包括控制生命期中发生的正常

变更，以及因需求变更而发生的变更。正常变更通常通过一个电子表格进行跟踪，它列出所有必须被变更的条目以及每个条目的目录（从而给出它的状态）。

如果项目经理允许并发地更新程序，则他们必须规定协调程序。并发更新有时是需要的。例如，假定正在实现一个变更申请的同时出现一个级别更高的变更申请。显然，在第一个变更申请完成之前，不能执行新的变更申请。同样，如果正在实现一个变更申请时工作系统出现了一个问题，则变更必须立即完成，以确保系统能够继续工作。对于这些情况，需要协调过程来解决。

一种可能的协调程序是表明将检查原版本和新版本之间差别，并且变更量更少的版本变更将被合并到其他版本中。如果变更影响程序的不同部分，程序合并是显而易见的；一些 CM 工具支持这些合并。否则，程序员必须检查重叠部分，然后协调这两种变更。

计划的所有要素都被记录在 CM 计划文档中。本章后面将给出 ACIC 项目的 CM 计划。

9.2.2 执行配置控制

虽然配置控制任务是在项目执行阶段承担的任务，但我们在这里与 CM 的其他特征一起讨论它们（配置控制任务）。配置控制任务主要有两个：一个涉及程序（和文档）的状态转移管理，另一个涉及必须被实现的变更申请的管理。

状态转移管理涉及在状态变更时将配置项从一个目录移到另一个目录，然后在完成变更后建立新版本。

通常用工具来管理项目的状态和版本以及对它们的访问。很多 CM 工具采用注册（check-in）/检查（check-out）方法进行访问控制和版本控制处理。这种方法的基本思想如下：当一个程序处于其他程序能够使用它的任何状态下，则认为该程序处于一个被控制的环境（controlled environment）中。一旦一个程序进入了被控制的环境，在没有正确授权的情况下就不能修改它，即使最初的作者也不能修改它，因为其他人有可能正在使用它。为了执行已批准的变更，开发人员必须在被控制的环境外面检查程序。检查实质上表示在不破坏早期版本的情况下制定配置项的一份拷贝，并标明配置项已经被检查过。

配置项经过检查以后，再对它进行修改。然后必须在被控制的环境中反映出配置项的修改，以便能够真正实现变更申请（它可能会强制要求这些变更）。因为

其他团队成员可能正在使用这些条目，所以在条目重新回到被控制的环境中之前，要对条目进行一些检查，以确保变更后的条目是合适的。当一个条目重新回到被控制的环境中时，旧版本的条目被破坏；而建立一个新版本的条目。通常，只有CC或者项目领导才能注册条目。这一限制使得在需要时能够撤消变更。

为了提供有关已作出的变更方面的信息，项目经理可以考虑为程序的源代码本身保持一个变更日志。这个日志实质上标识出一个变更的开始日期和结束日期，并包括一个对变更申请的引用。

所有这些任务——注册、检查、版本维护和变更日志的建立——都可以用相应的CM工具进行处理。可以用各种各样的工具来执行很多该CM库的功能。

为了实现需求变更，进而触发配置项的变更，首先通过执行影响分析（参见第3章讨论）而对变更申请进行分析。该分析确定需要进行修改的程序和文档，以及作此修改所需的成本及其对进度的影响。一旦项目领导和CC同意了此修改，所有在影响分析中标识出的程序和文档都必须作相应的修改。下面给出了实现一个变更申请时会牵涉到的一些任务：

- 接受变更申请（影响分析之后）
- 建立一种跟踪机制
- 检查需要进行变更的配置项
- 执行变更
- 注册配置项
- 在项目的整个生命期内维护该项目

通常用一种基于电子表格机制来跟踪变更申请的状态。对于每个变更申请，建立一个电子表格，列出所有需要进行变更的程序以及它们的状态。为了实现一次变更，CC或者CCB将不同条目的变更任务分配给团队成员，由他们检查项目并完成这些变更。在团队成员完成变更以后，可以将变更后的程序（或者文档）看成一个新的程序（或者文档），在它能够成为最终投产的系统的一部分之前必须经历不同的阶段（表示程序的生命期）。在所有修改后的程序及其相关的文档都到达了基准（在遵循它们的生命期之后），则认为变更申请已经被完全实现了。

9.2.3 状态监督和审计

一个配置项可以存在多种状态。状态集随配置项是程序还是文档以及所用的

CM 工具类型而变化。正确地表示每个配置项的状态是重要的，因为与状态相关的错误会引发一些问题。例如，如果一个程序还没有进行单元测试但是却进入了“准备发行”状态，则会出现问题。同样，如果系统不能反映出已经从基准库中检出一个程序并对它实施了变更这个事实，则发行的软件可能没有包含此变更。因此，在实现了一个需求变更申请后，确保用来获取配置项状态的机制正确地表示了配置项的状态，这也是很重要的。

如果项目使用一种基于目录结构的机制来表示一个程序的状态，则发生错误是可能的。这种机制要求在程序状态变化时将程序从一个目录移到另一个目录，并在维护各条目状态的主表中反映出状态变更。为了使错误最少并尽早地捕获任何错误，项目必须定期对配置项进行状态检查。可以建立一个关于差异的报告，所有这样的差异都必须被解决。

除了检查条目的状态外，项目还必须检查变更申请的状态。为了实现此目标，必须检查自最后一次 CM 状态监督以来收到的变更申请。对于每个变更申请，将变更申请记录中提到的条目的状态与实际状态进行比较。也可以通过检查来确保所有修改后的条目在回到基准库中之前都经历了它们的整个生命期(即，状态图)。

最后，项目可以执行一个配置审计。与其他审计一样，这里的主要目的是确保项目执行确实遵循 CM过程。也可以审计系统的基准库，以确保其完整性没有被违反，并且条目的移入和移出与CM 计划一致。

通常由项目的配置控制人员执行 CM 审计。CM 审计后通常发布一个报告，列出为了保持CM 任务与 CM 计划一致需要完成的任务。表 9.1 例举了一个 ACIC 项目的 CM 审计报告（该 ACIC 项目的 CM 计划在本章后面给出）。从表中可以看出，除了检查过程外，CM 审计主要关心配置项的状态和位置。

表 9.1 ACIC 项目的 CM 审计报告

序号	检查	负责人	完成日期
1	邮件被注册到 Messages/User 文件夹中	PL，DV	2000年 8 月 6 日
2	评审后的顺序图移入 VSS 下	PL	2000年 8 月 6 日
3	计划更新CM，改变目录结构。	CC	2000年 8 月 5 日
4	已完成的顺序图从用户区域移入评审区域	PL	2000年 8 月 6 日
5	更新 VSS 以反映存储结构的修改	CC	2000年 8 月 6 日
6	清除 Users 文件夹下的 ProjectDoc 子文件夹	所有人	2000年 8 月 6 日

9.3 ACIC 项目的配置管理计划

这里介绍的CM计划规定CM环境和ACIC项目所遵循的目录结构。(我以前的书中给出了另一个案例研究的CM计划。)ACIC项目的CM计划表明了目录结构在项目区域下，由Visual Source Safe（VSS）工具管理，所有需要被管理的文档都保存在这里。源代码文件由Visual Age for Java（VAJ）工具管理。在VSS目录下，规定了各种各样的目录：目录名明确地表明了这些目录将保存些什么。用户区域未受管理，但也对它进行了规定：每个用户一个目录，并且每个团队成员应该遵循其区域的目录指南。同样，也对评审区域进行了规定：这是保存要被评审的项目的目录。

接着，该计划列出配置项、它们的名字以及它们的存储位置。这里只列出受控的配置项。诸如测试结果、评审结果、消息、模板、标准等未受控的配置项都被忽略掉：它们分别被保存在各自未受控区域的目录中。对于每个配置项，在工作区域中给出开发时应当保存的区域。如果将要对配置项进行评审，则计划表明应当把它移到哪里进行评审，以及评审后它应保存的VSS下的基准区域。换句话说，计划概括了这些项目演变过程的存储区域。

再接着，计划描述了一个文档如何在这些不同的区域中移动——即，文档的配置控制过程。这是非常明确的：用户操作工作区中的文档。当文档准备评审时，就把它移动到指定的评审区域。如果评审不要求任何返工，文档就成为了基准。计划没有为代码规定类似的过程，因为使用了VAJ方法。

再接着，计划规定各区域和VSS的访问权。因为该项目的团队规模很小，所以遵循了一种相对较自由的访问机制，所有团队成员对受控区域都有注册和检查访问权。

再接着，计划规定变更控制过程。首先，计划规定谁负责记录变更申请、谁评审等等。然后给出变更实现过程。

协调是很多项目中的一个主要问题。前面已经提到过，当多个人同时修改一个配置项时，就需要进行协调。对于文档，不需要这样的协调，因为没有设计并行变更，并且VSS有正式的注册和检查程序，不允许并行变更发生。然而，源代码的变更协调是需要的。在类层次上，协调的责任由类的所有者负责，他将对此使用VAJ功能。(如果多个人对类进行了修改，VAJ强调类的不同和不同版本的方法。)在每个里程碑处，或者每当需要的时候，为所有资源进行所有协调。

如果在开发地点（脱机）和部署地点（联机）同时作出某些变更，这时可能需要进行协调。如果这种情况发生了，还是要使用 VAJ：标识出这些变更存在的差异，然后合并这些变更。

如果多个项目正在使用某些共同的源代码，则项目间的协调也是必须的。通常，所有项目将它们的 VAJ 文件发送到一个中央协调程序，由它协调文件并返回它们。然后，这些经过协调的文件就成为了每个项目的基准。

接着，CM 计划规定发行区域和后备过程。源代码的发行区域是 VAJ 仓储，并且在构造里程碑结束时发行源代码。（“发行”在这里表示协调已经完成，并且基准已经建立，这个基准也有可能发送给用户。）概要设计（high-level design，简称 HLD）在相应的里程碑结束时从 VSS 区域发行。对于后备，还规定了其区域和频度。然后，提到配置审计的性质和频度，以及配置控制人员的作用和责任。在这个项目中，CC负责维护工具、确定后备过程、执行审计并帮助团队遵循 CM 过程。（完整的 CM 计划还包含一些这里没有提到的要素。）

ACIC 项目的配置管理计划

1. 介绍

略。

2. CM 环境

- 操作系统：服务器为 Windows NT，PC 机为 Windows 98
- 其他软件/工具：MS Project 4.0，Rational Rose，Requisite Pro
- CM 工具：文档使用 Visual Source Safe(VSS)，源文件使用 Visual Age for Java（VAJ）

3. 目录结构

项目区域：Itlkec02/ACIC。所有目录都位于该目录下。

受控的文档存储区域：Itlkec02/ACIC/vss。在这个受控的区域，有诸如 HLD、ProgSpecs、ProjectDocs、ProjectMgmt、Requirements、Scope、TestPlans 等目录。

源代码：InfosysKEC/C://ivj.dat（VAJ 保存源文件的文件）。

未受控的项目区域：除了受控的 VSS 目录外，还有其他不同的目录，诸如 ChangeRequest、ImpactAnalysis、Issues、MilestoneReport、ReviewReports、StandardsAndChecklist、StatusReports、Templates 等等。这些目录都没有受到控制。

用户区域：不同用户的区域将是 ACIC/Users/<UserID>。每个用户将遵循受控区域的结构。

评审区域：ACIC/Review。这个区域下含有不同的目录，诸如 ProjectDocs、ChangeRequest、TestPlans 等等。

4. 配置项、命名和存储

这里只涉及受控区域的配置项。用标准的 Infosys 命名约定对所有文档和源文件进行命名。

配置项目	名字	工作区域	评审/测试区域	基准/发行区域
项目范围文档	ScopeDocument.doc			/vss/Scope
用例目录	UseCaseCatalog			/vss/Requirements
屏幕	Screens			/vss/Requirements
BAR 文档	BAR			/vss/Requirements
项目计划	ProjectPlan.doc	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/vss/ProjectMgmt
CM 计划	CMPlan.doc	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/vss/ProjectDocs
项目进度计划	ProjectSchedule.mpp	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/vss/ProjectMgmt
概要设计文档	TAD3.0.doc	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/vss/HLD
程序规格说明	ProgSpec#<n>	/Users/<用户名>/ProgSpecs/	/Reviews/ProgSpecs	/vss/ProgSpecs
单元测试计划	UnitTestPlan#<n>.doc	/Users/<用户名>/TestPlans/	/Reviews/TestPlans	/vss/TestPlans
顺序图	<描述>SeqDiag	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/VSS/RoseElements/SequenceDiagrams
类图	<描述>ClassDiag	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/VSS/RoseElements/ClassDiagrams
活动图	<描述>ActDiag	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/VSS/RoseElements/ActivityDiagrams
源代码	PackageName.Classname	VAJ 仓储	VAJ 仓储	VAJ 仓储
集成测试计划	IntegrationTestPlan	Users/ProjectDocs/TestPlans	/Reviews/TestPlans	/vss/TestPlans
测试计划	/UseCase#.tst	/Users/<用户名>/TestPlans/	/Reviews/TestPlans	/vss/TestPlans
收尾报告	ClosureReport.doc	/Users/<用户名>/ProjectDocs	/Reviews/ProjectDocs	/vss/ProjectDocs

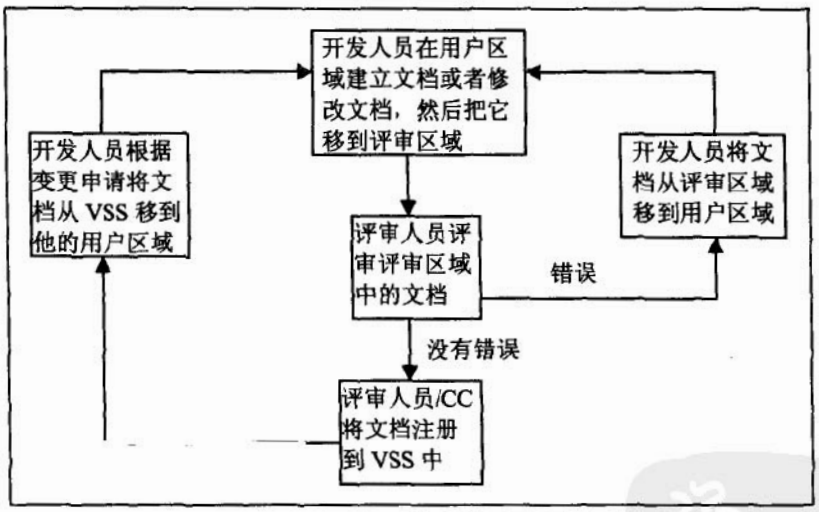
5. 版本/修订版编号

对于软件源文件：程序源文件有一种自动版本编号机制。临时版将是 1.0。源文件的任何主要修改将分配版本号 1.1、1.2等，而次要修改将分配版本号 1.1.1、1.1.2等。

对于文档：初始版本将编号为 0.0a。以后的修订版将编号为 0.0b、0.0c 等等。基准版本将是 1.0。文档可能会因重新设计或者用户变更申请而发生变化。新建立的版本将编号为 1.1、1.2 等等。

6. 配置项在存储区域间的移动

文档的配置控制过程如下图所示：



源代码的配置控制过程：源代码在VAJ 中进行评审。如果要对代码进行任何修改，则在用户正在工作的版本中进行。

访问权

用户区域：每个用户对其区域有 R/W 访问权。项目领导对所有区域都有 R/W 访问权。

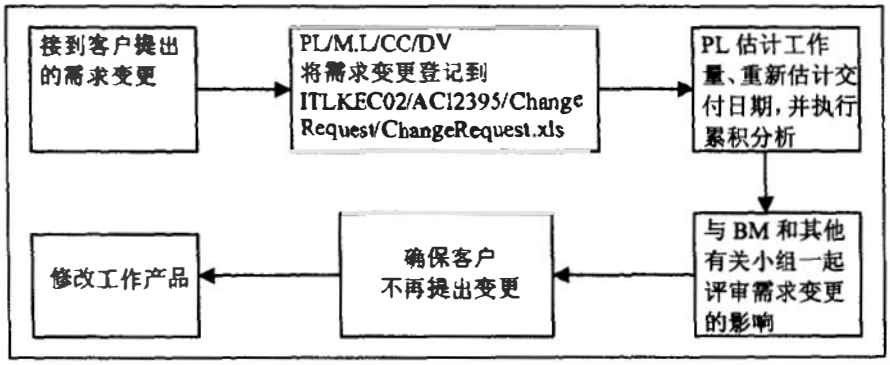
评审区域：所有人都有 R/W 访问权。

VSS 区域：所有人都有注册和检查权。

变更控制

变更申请登记在哪里?	ChangeRequest.xls
谁登记变更申请?	任何一个团队成员
谁评审变更申请?	项目领导
谁批准变更申请?	项目领导或者业务经理

修改申请的工作流



协调文档

所有文档都保存在 VSS 中。每次只允许检查一个文档。

源代码

联机/脱机协调：参考有关 VAJ 设置与协调（VAJ Setup & Reconciliation）方法的论文。

项目间协调：只要有源代码发行，就必须进行项目间协调。

项目内部协调：类级协调将是类所有者的责任。协调日期将在周项目会议上确定，并将使用 VAJ 的协调指南。

发行

配置项	发行区域	建立发行版的责任与发行的责任	何时发行
源代码	VAJ 仓储	工作空间所有者	里程碑分析结束时
HLD	vss/HLD	PL	概要设计阶段结束时

备份

备份区域	备份介质	编号机制	备份	
			频度	责任
Itlkec15/	磁带	根据 Infosys 标准	每周	系统管理员
VAJ 仓储（服务器的 ivj.dat）	ITLKEC02\\\\ProjectBackup	mmddyyyy.dat	每周 2 次	CC
VAX.icx 和.ide 文件	ITLKEC02\\\\Users\\<用户名>\\General	mmddyyyy.ide 和 mmddyyyy.icx	每天	用户

存档过程

存档由 Infosys 系统管理人员用标准过程完成。

配置审计

审计类型	频度
CC 审计	每 2 周 1 次
基准审计	每月 1 次
工作产品完成	里程碑结束时
发行	发行前
意外事件审计（Surprise audit）	随时

配置控制人员的责任

- 项目团队面向CM
- 如果有CM工具，则负责部署CM工具
- 配置项状态跟踪
- 保证备份和存档
- 根据计划执行 CM 审计
- 产生 CM 审计报告
- 跟踪CM审计差异，直到结束

9.4 小 结

软件产品通常包括很多程序和文档，在这些项能够成为最终系统的组成部分之前，它们还会发展变化。因此，软件配置管理是一个重要的问题。

下面列出了从 Infosys 公司遵循的 CM 过程中吸取的一些经验教训：

- 定义 CM 过程，使项目能够处理并发更新、撤消一个变更、获得一个程序的最新版本、确定一个程序的状态，以及防止未授权的变更。为了支持这些功能，应使用版本控制、变更申请跟踪以及库管理机制。
- 开发一个独立于项目管理计划的 CM 计划。CM 计划必须规定环境、配置项及其命名约定、项目在不同状态的存储区域，以及管理项目变更的方法，包括版本编号和协调、访问控制以及发行和备份策略。
- 执行 CM 审计和状态检查，确保 CM 计划得以遵守。

关于 CMM，本章讨论的任务满足 CMM-2 的软件配置管理 KPA 的很多需求。需求变更的实施方法满足需求管理 KPA 的一些需求。工作产品（而不是代码）基准化和控制也满足诸如需求管理和软件项目规划等 KPA 的基准化需求。

9.5 参考文献

[1] E. H. Bersoff, V. D. Henderson, and S. G. Siegel. Software configuration management: A tutorial. *IEEE Computer*, Jan. 1979.

[2] E. H. Bersoff. Elements of software configuration management. *IEEE Transactions on Software Engineering*, Jan. 1984.

[3] W. Humphrey. *Managing the Software Process*. Addison-Wesley, 1989.（"SEI 软件工程丛书"之《软件过程管理》，瓦茨·汉弗莱著，其英文影印版由清华大学出版社于 2002 年出版，中译版由清华大学出版社于 2003 年出版。）

[4] D. Whitgift. *Methods and Tools for Software Configuration Management*. John Wiley and Sons, 1991.

[5] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys*. Addison-Wesley, 2000.

第 2 部分　项目执行和收尾

第 10 章　评　　审

你可以通过测试检查一个程序的质量。但是如何检查测试采用的测试计划有正确的测试案例呢？如何检查设计中存在的设计错误或者需求规范中存在的不足呢？评审是最有效的也是最常用的标识故障的方法，它不仅可以用于诸如测试计划和设计文档等不可执行的文档，也可用于代码。评审还可以使管理人员掌握项目的进展，这有助于他们及时采取纠正措施。此外，项目经理还可以从评审中得到如下一些好处：

- 通过评审，项目可以利用组织中最有才干的人，即使他们没有参与该项目也能发挥作用。
- 评审令项目人员有一种成就感、参与感和得到认可的感觉，从而保持团队的积极性。
- 通过评审，团队成员可以发展他们的技能，而资深成员可以指导那些缺乏经验的成员。
- 通过评审更加留意故障，从而帮助预防故障。

本章讨论 Infosys 公司采用的评审过程，说明了如何收集数据及如何用它们来监督和控制这些评审。我们还简单地考虑了如何用实验使项目经理确信小组评审的价值。

10.1　评审过程

评审可以按不同的方法进行。一种正式的小组评审（group review），即所谓的检查（inspection），可能是这些识别故障的评审方法中最好的一种。软件检查最初由法根（Fagan）提出[1,2]，在这之前的检查主要考虑代码检查，但是几年后

这种技术延伸到了其他工作产品中。今天，软件检查是一种业界公认的最佳实践，有很多资料证明它们能够提高质量和生产率(例如，Gilb 和 Graham 的报告、Grady 和 Slack 的报告及 Weller 的报告，等等)。几本关于该主题的书详细地描述了如何执行检查。

Infosys 公司采用的基本评审过程是小组评审，这与检查相类似。小组评审是一组从事相同项目的人员遵循一个明确定义的过程对一种软件的工作产品进行分析。这种评审的目标是：通过发现故障而提高质量，通过有成本效益的故障查找而提高生产率。此外，评审提供了项目管理所需的数据，并可以监视工作产品的质量。小组评审由技术人员替技术人员执行，重点是标识问题，而不是解决它们。

小组评审过程包括几个阶段：规划、准备和概述、一次小组评审会议，以及返工和后续工作。如图 10.1 所示，这些阶段通常按线性顺序执行。

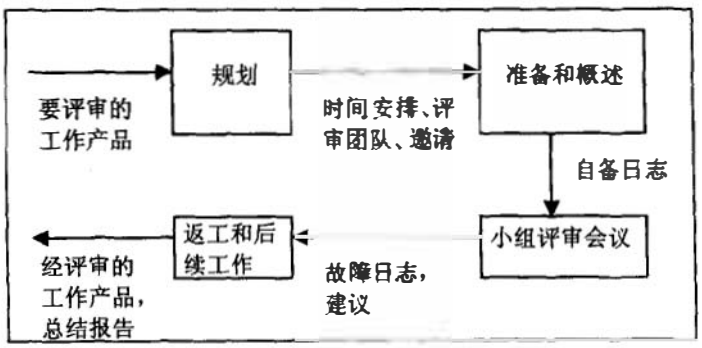


图 10.1 小组评审过程

在某些情况下，小组评审可能是一种过度行为；一种更加有限的小组评审形式可能会更加有成本效益。在 Infosys 公司，还要执行个人评审，这将在本节后面进行讨论。

10.1.1 规划

规划阶段的目标是为小组评审作好准备，选择小组评审团队和安排评审时间。工作产品的作者保证准备好工作产品进行小组评审，并且满足所有相干的标准。项目经理在作者的同意下，首先选择仲裁人（moderator）；然后与仲裁人协商，选择其他评审人员。仲裁人对评审负全部责任，既要保证评审正确地完成，又要

保证遵循评审过程的全部手续。评审人员（reviewer）（也称为检查人员）负责标识工作产品中的故障．通常，作者也是评审人员之一；仲裁人也可以是一个评审人员。

尽量不要有上级人物参加，因为他们在场可能会妨碍评审人员提出问题或者错误。然而，如果作者愿意，项目领导也可以参与评审。

一旦选定了评审团队，作者准备一个分发给小组评审的软件包。该软件包包含要被评审的工作产品、其规范以及有关的检查表和标准。通常，工作产品的规范是先导结果，是检查当前工作产品的正确性所需的。

10.1.2 概述和准备

概述和准备阶段的目的是将要被评审的软件包交给评审人员，并在需要时对工作产品进行说明。材料可以在第一次会议上发给评审人员并加以说明。仲裁人主持这次会议，简单介绍工作产品、小组评审目标，并在需要时概述一下小组评审过程。作者可以提供一份关于工作产品的简要指南，总结所有特殊的考虑或者有可能特别难于理解的领域。概述时，任何有关项目或者工作产品的特殊之处都要加以强调。这一步是可选的，可以忽略它。如果不考虑这一步，仲裁人只是将小组评审包的拷贝发给评审人员。

为了准备小组评审会议，各评审人员独立地评审工作产品（自我评审）。只要认为工作产品可能有故障，评审人员就在一个自备日志上加注说明（自备日志的详细描述见 10.2.1 节）。他们还记录单独评审所花的时间。在这评审期间，他们要使用有关的检查表、指南和标准。

理论上，该准备手续应当在一次会议上完成．建议的准备时间是：为每个小时的小组评审会议安排 1 个小时的准备时间。

10.1.3 小组评审会议

小组评审会议的基本目的是：拿出最终的故障列表；该列表基于评审人员报告的故障和问题的初始列表，以及会议讨论期间发现的任何新的故障。

在安排会议之前，仲裁人首先检查所有评审人员是否都已准备好。该查证涉及到简单地检查自备日志中的工作量和故障数据，以证实准备工作已经投入了充

分的时间和精力。如果准备工作不充分，则小组评审推迟到所有参与者都准备好为止。

当一切都准备就绪以后，举行小组评审会议。指定一个评审人员担当书写员，另一个担当阅读员。会议如下举行。阅读员逐行（或者根据任何其他合适的小单位）阅读工作产品，向团队解释每一行内容。（某些会议不用解释，因此可以不设阅读员这个角色。）在任何一行，如果某个评审人员在独自评审时标识出了某些问题，或者在会议上听他人阐述时发现了新问题，他就要提出该问题。可能要对提出的问题进行讨论，而其他评审人员可能同意其意见也可能不同意。无论如何，作者要检查正在讨论的问题，要么阐明它为什么不能算一个问题，要么接受提出的问题，承认它是一个故障或者一个未解决的问题。书写员记录标识出的问题和故障。

会议结束时，书写员上交会议期间标识的未解决的问题和故障的列表，并且该表还要经过团队成员的最终评审。注意：评审过程只是标识出故障和问题。其解决方案不是小组评审的责任；而是由作者以后采取解决问题的措施。

如果几乎没有要求什么修改，则小组评审状态为“已接受的”。如果要求的修改很多，则可能有必要举行后续会议或者重新评审，以验证是否已经正确地实现了变更。仲裁人建议要采取什么措施。问题与故障的解决方法不同，故障由作者负责修复，而对于问题，仲裁人可以将它分配给其他人去解决。该任务分配在会议结束前完成。有时，还要给出关于下一阶段的评审的建议。例如，进行详细设计的评审时，小组可以建议哪些代码模块应当在构建阶段接受小组评审。

仲裁人负责会议，确保会议围绕基本主题，即故障识别，并且不会发展成一种自由讨论会或者个人攻击作者。仲裁人应当接受有关如何执行评审方面的培训，或者应当有过参与一些评审团队的经验。会议期间，仲裁人必须确保所有参与者都畅所欲言、确保能够听到每个评审人员的意见、确保对评审时发现的问题取得一致意见，以及确保大家的兴趣不下降。一个关键责任是确保会议围绕问题识别这个主题不变，而不会变成问题解决会议。总之，维持会议有序和友好主要是仲裁人的责任。会后，仲裁人必须确保所有参与者都感到满意，确保评审报告已填好并且通知下去，确保已经采取了后续措施。

在评审会议上，一个人可以担当几个逻辑角色，但是作者不能担任仲裁人或者阅读员，并且仲裁人不能是阅读员。这种限制意味着小组评审团队起码要有 3 个人：作者、仲裁人和阅读员。这 3 个人也是评审人员，并且其中 1 人可以承担书写员。

10.1.4 返工和后续措施

作者执行返工，以改正小组评审会议上提出的所有故障。如果仲裁人建议作者重做工作产品，则作者可能不得不重做。此外，如果评审人员被分配了未解决的问题，他们必须研究这些问题并将研究结果交给作者和仲裁人。

作者与仲裁人一起审查改正情况，如果仲裁人决定必须再次举行评审会议，则在重新评审中审查改正情况。书写员保证将小组评审报告和会议记录通知给小组评审团队。在所有问题和故障都解决后，仲裁人确保小组评审结果和数据都被记录下来，并将小组评审总结表（参见 10.2.3 节）递交给 SEPG 和项目领导。

10.1.5 单人评审

小组评审是一种非常有效的识别故障的方法。遗憾的是，其成本也较高：不仅很多人要在准备上花时间，而且在会议上也要花时间。此外，安排小组评审会议也是一个复杂的逻辑问题，相应地也要一定的成本。如果工作产品有很多故障或者是一种急待交付的产品，则付出此成本是合理的。小组评审确实是一种有效地发现故障的方法，但是，如果工作产品相对比较简单、不可能有很多故障，并且也不急于交付使用，这时小组评审的成本效益比如何呢？在这种情况下，小组评审的努力可能会得不到回报。然而，对这些工作产品执行某种评审是有用的，这不仅可以检测到故障，而且还能达到心理上的作用，因为当作者知道还有他人将评审其产品时，他会更加重视该产品。

在这种情况下，采用单人评审可能会更加合适。即，对于紧急程度或者复杂程度为中等的工作产品，可以用单人评审取代小组评审。单人评审（one-person review）是正式评审，但是比小组评审所需的成本更低，因为他们不用组成一个评审团队。

单人评审的过程类似于小组评审过程．作者征询项目领导的意见，确定评审人。他还要安排好评审时间，让评审人提前收到评审软件包。评审人单独地评审工作产品，并作好与作者举行会议的准备。评审会议只有两个参与者：作者和评审人。会议期间，建立一个问题日志和一个故障日志。评审结束时，评审人通知项目领导。项目领导负责跟踪故障，直到项目收尾。

10.1.6 项目评审指南

并非一个项目的所有工作产品都要经过小组评审，因为这样做可能要付出很高的成本，并且有可能得不到相应的回报。下面给出了选择要进行评审的工作产品的指南。

对于一个特定的项目，有关工作产品要被评审的实际条件和决策，由项目经理和早期阶段工作产品的评审团队负责。有了早期工作产品的评审结果后，团队就能更好地决策项目后期要评审哪些产品以及采用哪种方法。因为项目生命期早期的工作产品非常关键，并且因为它们所含的故障在后期阶段有着放大器效应，所以建议对如下工作产品进行小组评审：

- 项目管理计划
- 需求规范
- 系统测试计划
- 概要设计
- 集成测试计划

概要设计评审结束时，小组评审提出详细设计的评审建议；而在详细设计评审结束时，为代码评审提出类似的建议。

尽管评审过程对于任何一种工作产品都是相同的，但基于工作产品的性质，评审过程也会因评审重点、评审开始条件和评审团队构成而出现差异。所用的检查表也取决于工作产品的性质。表 10.1 概括了几种工作产品的评审指南。其他工作产品的评审指南与此类似。

10.2 数据收集

评审期间的数据收集非常重要。你已经看到，在评审过程的各个阶段，数据都要被记录下来。因为评审主要是人为过程，所以，如果数据没有被正确地记录下来，信息很容易被丢掉。除了记录评审过程识别出来的故障外，还必须记录工作量数据。详细的故障数据是跟踪项目故障所必须的。总的故障和工作量数据是分析评审的有效性和构造评审能力基准所必须的。因此，每次评审的总结数据都被保存到一个评审数据库中。本节描述 Infosys 公司在评审期间用于数据收集的主要形式。

表 10.1 工作产品评审指南

工作产品	评审重点	评审开始条件	参与者
需求规范	● 需求满足客户需要 ● 需求是可实现的 ● 需求中遗漏的、不一致的和不明确的内容	● 文档符合标准	● 客户 ● 设计人员 ● 测试人员（系统测试） ● 安装团队成员 ● 用户文档作者
概要设计	● 概要设计实现了需求 ● 设计是可实现的 ● 设计中存在的疏漏和其他故障	● 文档符合标准 ● 需求已经被评审过，并且被最终定下来	● 需求作者 ● 详细设计人员 ● 开发人员
代码	● 代码实现设计 ● 代码是完备的和正确的 ● 代码中存在的故障	● 代码编译和扫描风格，以及其他规范	● 设计人员 ● 测试人员 ● 开发人员
系统测试用例	● 测试用例集能够检查需求中的所有条件 ● 系统测试用例是正确的 ● 测试用例是可执行的	● 需求已经成为基准 ● 系统测试计划与标准一致	● 需求作者 ● 测试人员 ● 项目领导
项目管理计划	● 项目管理计划满足项目管理和控制需要 ● 完备性 ● 项目管理计划是可实现的 ● 遗漏的或者不明确的内容	● 项目管理计划遵循标准模板	● 项目领导 ● SEPG 成员 ● 其他项目领导

10.2.1 自备日志

评审人员使用自备日志（self-preparation log）来记录他们单独评审期间发现的所有故障或者问题；同时还要记录评审所花的工作量。每个评审人员都要准备这个日志。除了标识项目、工作产品、评审人员和其他事实外，日志中的记录项还规定问题或者故障的位置，以及评审人员指定的严重性和紧急性。图 10.2 所示的表单可用作自备日志，并且可以通过一个电子表格来实现。

10.2.2 小组评审会议日志

书写员准备一个小组评审会议日志，列出会议期间标识出的故障和问题。因此，该日志包括各评审人员单独评审时发现的并且在会议上被确定为故障或者问题的所有故障，以及在会议期间发现的新故障。小组评审日志与自备日志不同，后者列出评审人员检查到的故障，而前者只列出那些经作者认可的故障。

项目代码：
工作产品ID：
评审员名字：
评审准备所花的工作量：
问题列表

SI#	位置	描述	紧急性/严重性

图 10.2 自备日志

换句话说，小组评审日志列出发现的真实故障，是正式的评审故障列表。

除了记录每个故障的位置和描述外，该日志还记录故障的严重性。严重性反映了小组评审团队的共识。在第 7 章已经讨论过，故障的严重性可能是严重故障、主要故障、次要故障或者装饰性故障。如果能够确定故障是在哪个阶段引入的，那么也要记录此信息，如果可能的话还要记录故障的类型。

提出的问题列在一个单独的日志中。对于每个问题，分配一个人去解决它。该日志还包含会议所花的总工作量（人时）。

图 10.3 表示了小组评审会议日志的格式。这可以通过电子表格或者其他某种机制来实现。

10.2.3 小组评审总结报告

故障日志用来跟踪所有要解决的故障。然而，为了分析评审的有效性，只需总结级信息。该信息也是更新评审基准所需的。对于评审过程的过程改进和理解，小组评审总结报告是最重要的要素。因此，该信息保存在一个独立的评审数据库中，分析时可以使用它。

总结报告描述工作产品、所花的总工作量和评审过程每个任务所花的工作量、发现的每类故障的总数、以及正被评审的工作产品的规模。如果记录了故障的类型，则可以在总结报告中记录每类故障的故障数。除了关于工作量和故障的数据外，该总结报告还包含下一阶段的建议。最后，总结报告指明是否需要重新评审。本章后面给出了一个已完成的总结报告。

项目代码：　　会议类型：
仲裁人：　　书写员：
作者：　　评审人员：
SEPG成员：　　见证人：
日期：　　工作产品ID
评审会议所花的工作量（人时）：
要被解决的故障（日期）：
故障列表

SI#	故障位置	类型	严重性	故障引入阶段	描述

未解问题的日志

SI#	问题描述	负责人	计划的日期	解决日期

图 10.3 小组评审会议日志

10.3 监督和控制

评审过程的有效性取决于过程部署得有多好。例如，如果在评审一个 500 行的程序或者一个 20 页的设计文档时只发现 2 个故障，那么评审显然是无效的。评审效率不高的最常见的原因是没有遵照评审主题和采取严肃的态度进行评审。除非严肃地对待评审，否则它们很可能是对时间的巨大浪费，并且不会得到任何应有的回报：否则会被人认为随随便便地执行评审就可以了。

项目经理或者仲裁人如何评价一次评审是否有效，以便她能够决定未来的行动方针呢？一种有效地监督和控制评审的方法是采用通过控制图实现的统计过程控制（SPC）概念。因为评审的数据点数量可能非常巨大（尤其是代码评审），所以可以满怀信心地应用统计技术。本节讨论 Infosys 公司如何用统计技术监督和控制评审。

10.3.1 评审能力基准

如何才能应用 SPC 监督评审呢？为了应用 SPC，项目经理必须标识出关键的绩效参数，确定它们的控制极限，然后监督它们的实际绩效。他们可以通过绘制评审的绩效参数建立控制图，然后用此图评价评审的有效性。另一种方法是设置各种参数的控制极限，然后用此范围确定评审的有效性。虽然后一种方法存在一定的局限性，因为运行图不容易得到，但是它们用起来很简单。在 Infosys 公司，评审有效性的评价采用后一种方法。

在 Infosys 公司，确定了如下绩效参数的控制极限：准备期间的评审速度、小组评审会议期间的评审速度、次要故障或者装饰性故障的故障密度，以及严重故障或者主要故障的故障密度（总的故障密度只是前两种故障密度的和）。这些极限根据以往的数据和评审能力基准而确定。建立和维护该基准是收集有关评审的总结数据的重要原因。表 10.2 表示了小组评审能力基准。

表 10.2 Infosys 评审能力基准

评审项	准备期间的评审速度（如果不同于小组评审期间的评审速度）	小组评审期间的评审速度	装饰性故障/次要故障的故障密度	紧急故障/主要故障的故障密度
需求	5-7 页/小时	0.5-1.5 个故障/页	0.1-1.3 个故障/小时	
概要设计	4-5 页/小时(或 200-250 规范语句/小时)	0.5-1.5 个故障/页	0.1-1.3 个故障/小时	
详细设计	3-4 页/小时（或 70-100 个规范语句/小时）	0.5-1.5 个故障/页	0.2-0.6 个故障/小时	
代码	160-200 LOC/小时	110-150 LOC/小时	0.01-0.06 个故障/小时	0.01-0.06 个故障/小时
集成测试计划	5-7 页/小时	0.5-1.5 个故障/页	0.1-1.3 个故障/小时	
集成测试用例	3-4 页/小时			
系统测试计划	5-7 页/小时	0.5-1.5 个故障/页	0.1-1.3 个故障/小时	
系统测试用例	3-4 页/小时			
项目管理和配置管理计划	4-6 页/小时	2-4 页/小时	0.6-1.8 个故障/小时	0.1-1.3 个故障/小时

表 10.2 的小组评审基准给出了各种工作产品在准备期间的评审速度、小组评审期间的评审速度以及紧急故障或者主要故障的故障密度（总的故障密度只是这两种故障密度的和）。故障密度根据规模进行了规范化，而非代码性工作产品的规模用页数度量，对于代码性工作产品用代码行度量。（对于设计，规模也可以根据规范语句的数量进行度量。）评审速度根据每单位工作量的规模来表示，其中的工作量以人时进行度量。你可以看到，对于文档，评审速度和故障密度非常接近，但是对于代码它们却是不同的（其中的规模单位也是不同的）。

单人评审的评审速度应该是不同的。详细设计文档、测试计划和代码一般接受这种评审形式。因此，制定了这些工作产品的单人评审基准。对于文档的单人评审基准，每小时的评审速度大约是相应的小组评审的评审速度的 2 倍；每页的故障检测率大约是小组评审发现的一半。对于代码，每小时的评审速度基本相同，但是每 LOC 的故障检测率大约少 30%。

该基准是监督一个项目执行的评审的基础。

10.3.2 分析和控制指南

基准中给定的范围用于确定评审的绩效是否在可接受的极限内。该检查被规定为评审过程的结束标志。项目经理可以将结束标志定义为检查所有不同的参数是否在范围内，但是因为检测故障是评审的主要目的，所以结束标志是总的故障密度必须在规定的极限内。（另一种方法是检查两种故障类型的故障密度都在适当的范围内。）如果评审期间发现的故障数在基准给定的范围内，则认为评审是有效的，结束标志被满足，并且该评审不需要进一步的措施。

如果不用评审能力基准，项目经理可以用一个内部开发的SPC工具监督评审。该工具实质上是一个内嵌了能力基准数据的电子表格。它还有关于故障引入率、故障排除效率、组织范围的质量能力，等等数据。根据这些数据，该工具确定评审的绩效规范——即，在满足质量目标的情况下，其结果应该满足的范围。当某个评审数据不在控制极限或者期望的极限之内时，SPC 工具就发出警报。

如果评审时发现的故障密度不在能力基准给定的范围内，则不能机械地认为评审是失败的。项目经理或者仲裁人要严格地评价情况，并决定下一步应采取的措施。这里的准备速度和评审速度非常有用：如果评审速度与基准中给定的速度相比“太快”了，则评审效率不高的原因相对比较明确。次要故障和紧急故障的

故障密度在此分析中也是有用的。尽管仲裁人员或者项目经理可以采用任何一种技术来确定出现绩效偏差的原因，以及应该采取的纠正和预防措施，但是表 10.3 给出的一些指南有助于这一个任务的完成。

表 10.3 评审的分析指南

可能的原因	应考虑的措施
如果发现的故障数少于标准	
工作产品非常简单	● 变类似工作产品的小组评审为单人评审 ● 两种评审（小组评审和单人评审）结合
评审可能不够全面	● 检查评审速度：如果评审速度太低，则重新安排一次评审，并且有可能要组织一个不同的团队
评审人员没有经过充分的小组评审培训，或者没有评审材料的经验	● 安排时间进行小组评审培训 ● 组织另一个团队进行重新评审
工作产品的质量非常好	● 根据评审速度、作者的经验、评审人员等指标确认该事实；弄清楚项目的其他部分是否有相同的质量 ● 修正后续任务的故障预测；看看有没有通用的过程改进教训
如果发现的故障数多于标准	
工作产品的质量较低	● 检查作者的培训需求 ● 重做工作产品 ● 考虑重新分配未来的任务（例如，仅为该作者分配更加容易的任务）
工作产品非常复杂	● 确保后续任务执行良好的评审或者测试 ● 增加系统测试的估计 ● 将工作产品分解成更小的组件
次要故障太多（而主要故障太少）	● 识别出导致次要故障的原因；通过适当地增强检查表并使作者注意常见的原因，在以后的工作中纠正这些故障 ● 评审人员对工作产品理解不够。如果这样，举行一次综述会议，或者让其他评审人员再进行一次评审
评审的参照文档不正确和不明确	● 评审参考文档，并使它得到批准
被评审的模块是项目的第一个模块	● 分析故障，更新评审检查表，并通知开发人员。安排培训时间

如果过程在一个稳定的状态下运作，则静态控制极限能很好地起作用。然而，如果有意识地对过程作出变更，则必须谨慎地使用控制图，因为过程绩效也会发生变化。如果过程变更有可能经常发生，则最好采用动态控制极限。一种方法是

根据以前的 n 个绩效数据点将控制极限重设为新值（n 必须被选定，可能是某个比 10-15 还要大的数）。用这种方法，如果过程改变了，其绩效也会改变，并且经过几个数据点之后，控制极限将反映出变更后的过程的绩效能力。

另一种方法是根据过程变更期望的影响调整绩效数据或者控制极限。例如，如果评审过程已改变，并且期望评审将多检测到 10%的故障，那么当一个点落在控制极限之外时，在分析时就必须考虑这一事实。

后一种方法是 Infosys 公司在评审期间进行故障预防时采用的方法。采用 DP（故障预防），故障引入率应该下降。因此，评审时检测到的故障密度也有可能下降。如果一个项目在使用 DP，那么在项目规划期间还要记录期望的 DP影响。使用故障引入率，就可以确定 DP 使用可期望的故障减少、以及故障检测率、期望的绩效极限。如果一个评审的绩效不在这些极限内，则应当仔细地检查其原因。

10.3.3 示例

考虑表 10.4 给定的项目管理计划的小组评审总结报告。该总结报告牵涉到一个有 14 页的项目管理计划的小组评审。小组评审发现的次要故障和装饰性故障的总数是 16 个，而发现的主要故障的总数是 3 个。因此，每页发现的故障密度是 16/14=1.2 个次要故障，以及 3/14=0.2 个主要故障。两种比率都在能力基准给出的范围内，因此结束标志满足，可以认为正确地实行了评审。

因为这次评审的结束标志已经满足了，所以不需要检查其他比率，但是在其他评审中可能还要检查其他比率。评审团队有 4 个成员，单人评审都花了 2.5 个小时；评审会议进行了 2.5 小时。因此，在准备和评审期间的评审速度是每小时 14/2.5=5.6 页，这个评审速度在准备期间的速度范围内，但略高于评审期间的速度范围。

当把总结数据送入 SPC 工具时，它就图形化地显示这次评审的绩效，以及控制极限和期望的极限。

表 10.4 评审总结报告

项目	
工作产品类型	项目计划，v.1.0
产品规模	14 页
仲裁人	米乐（Meera）
评审人员	Biju、米乐
作者	JC
工作量（人时）	
a.综述会议	0
b.准备	10 人时
c.小组评审会议	10 人时
总工作量	20人时
故障	
紧急故障数	0
主要故障数	3
次要故障数	12
装饰性故障数	4
准备期间检测到的故障数	-
小组会议期间检测到的故障数	
提出的未解决的问题的数量	1
总故障数	19
结论	仲裁人复审
下一阶段的建议	
接受小组评审的单元	N/A
接受单人评审的单元	N/A
备注（仲裁人）	计划用文档很好地记录下来，并且已呈送给项目领导
准备者：米乐；日期：xx-xx-xxxx	

10.4 评审和 NAH（这里不适用）综合症介绍

评审在很多方面都是反直觉的。程序员不明白如何才能使由一组人执行的评审比测试更加有效。当一个项目的最关键的资源是人力时，则很难让人接受人力高度集中的评审过程能够使整个过程更加有效，并能改进质量之类的观点。因此，使人们相信使用评审是最难的过程部署任务之一。SEI 报告指出：只有 22%的软

件组织采用了某种形式的检查。[8]

显然，硬数据对于证明这种情况是无价的。不少公布的数据支持评审是有成本效益的并能切实改进质量。但是，这些由世界各地的组织公布的数据不能使工程师相信评审对他们的组织是有益的。产生这种怀疑的一个原因是这里不适用（Not Applicable Here，简称 NAH）综合症：人们相信评审对其他组织是有益的，但是他们公司的情况不同，因此评审不适用。[9]

如果要在一个项目中实施检查，则必须克服 NAH 综合症。必须向管理人员及开发人员说明检查确实能够提供好处。根据定义，其他组织的数据不能用来克服NAH综合症。相反，必须使用本组织内部的数据来建立一个检查案例。因此，需要一种实验性部署计划，可以将它迅速地部署到现实环境中，以此评价检查是否适应组织。

在试图克服 NAH 综合症方面，Infosys 采用了一个简单的实验，关于该实验的描述见下文。这是一个通用而简单的实验，任何组织都可以轻而易举地实现它。有关此工作的细节，参见我与人合写的论文。[9]

10.4.1 Infosys 实验

Infosys 实验集中解决代码检查问题，因为开发人员与编码有关，并且代码往往是绝大多数错误的来源。在过去，检查从代码开始，直到后来才延伸到设计、需求、测试计划以及其他条目。一旦为代码检查建立了有力的案例，并且这些案例都被部署了，则这些检查的好处会变得明显起来，并可以建立其他工作产品的检查案例。

对于 Infosys 公司实施的实验，选定了 6 个关于某个银行业产品的系统增强申请（system enhancement request，简称 SER）。首先对开发人员进行了检查过程培训；作为练习，要求他们检查一个更早些时候实现的 SER，其中夹杂了一些故障。

全体人员被分成两组，每组 3 人。每组构成一个检查团队。在提交SER之前，要求每个开发人员实现 SER、编译其代码，并进行一些自我测试。提交之后，分别按两种方法独立地对 SER 进行查错：检查和单元测试。实验期间，一个检查团队检查该组成员部署的 3 个 SER 的代码。在每次检查中，作者（而不是仲裁人或者阅读员）担任检查者。同时，SER 所属领域的模块领导独立地对 SER进行单元测试。图 10.4 所示的流程图表示了基本实验步骤。

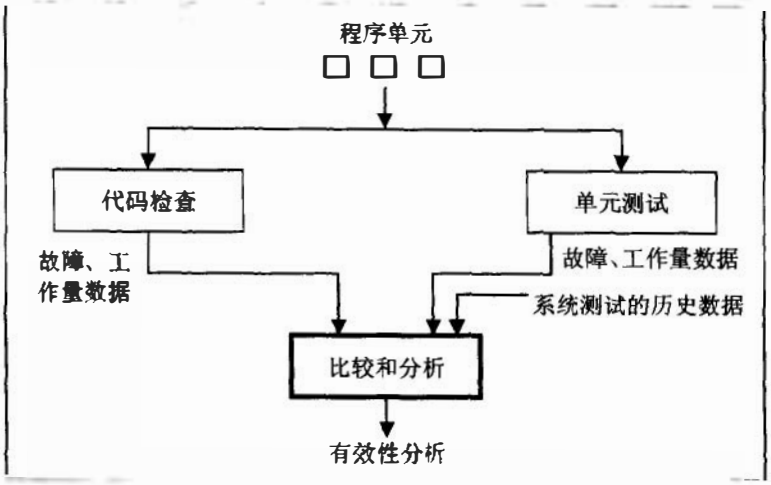


图 10.4 Infosys 实验的基本步骤

对于任何一种查错方法，都要记录其所花的工作量和发现的故障。然后，用这些数据来评价评审的成本效益和质量。如果这两种方法所发现的故障集合不同，并且如果一个故障集合不是另一个故障集合的子集，则能够声明检查方法确实发现了一个不同于单元测试所发现的故障集合，所以增加检查方法是有好处的。理解查错的成本要更难一些（也是引起大多数疑问的地方），一般根据以往的系统测试数据来评价差错的成本。

10.4.2 实验数据

表 10.5 给出了 SER 的规模、总的工作量以及两种查错方法所发现的故障数。显然，检查方法找出了比单元测试方法更多的故障。所有 SER 都一致地观察到了这一发现。总之，虽然检查方法比单元测试方法消耗的时间更多，但是检查方法找到的故障数是单元测试发现的故障数的 2.5 倍。然而，检查方法和单元测试方法检测到的故障数/人时是相同的；两者都大约为 1.9 个故障/人时。

现在，让我们看看这两种方法发现的故障的性质，如表 10.6 所示。几乎在所有分类中，检查方法都比单元测试方法检测到更多的故障，尤其是与诸如可维护性、可移植性等质量属性相关的故障分类（如果测试主要关心功能方面的错误的话，这种结果是可想而知的）。表中的数据还表明：即使在逻辑和接口故障方面（测试重点），检查方法也比单元测试方法更好。从这些数据中可以看出，增加检查方法显然可以提高错误检测能力，这个案例是有说服力的。

表 10.5 工作量和故障数据

规模 SER	检查			单元测试	
	总工作量		总故障数	总工作量（小时）	总故障数
	LOC	小时			
1	968	8.0	8	2.0	4
2	432	5.0	8	1.5	3
3	85	4.0	4	1.5	1
4	667	6.5	26	1.5	7
5	50	12.5	3	1.5	0
6	408	2.5	5	2.5	5
总计	2 610	27.5	54	10.5	20

表 10.6 故障分布

故障类型	检查	单元测试	常见故障数
数据	3	1	0
函数	4	2	0
接口	14	11	7
逻辑	12	5	4
可维护性	11	0	0
可移植性	5	0	0
其他	5	1	1
总计	54	20	12

这些数据表明每个故障的成本基本相同。但是，评审有更高的成本效益比。根据以往的经验和数据，我们知道系统测试期间找出并排除一个故障大约要花 4 人时。如果系统测试没有发现一个故障，则以后要花 2 人日（17 人时）才能识别和排除它。

测试几乎不能查出可维护性和可移植性类型的故障。我们假定单元测试没有查出的所有逻辑、接口和数据故障后来都被发现了。在表 10.6 中，这些故障的数量分别是 3、4、14 和 12。在单元测试消除了常见的故障后，那些故障的数量变成了 3、4、7 和 8。如果要在系统测试阶段查出所有这些故障，则系统测试的成本是 22×4=88 小时，或者说大约 11 人日。如果系统测试发现其中的 75%故障，而 25%的故障在后来查出，则系统测试要付出的额外成本是 0.75×22×4=66 小时（9.5 人日）；修复后来发现的故障的额外成本是 0.25×22×2.5=11 人日。即，如果不增加检查方法，则在修复额外的故障方面将多花 20.5 人日。

因此，如果要在系统测试时找到所有故障，则由于增加检查方法而产生的成本节约是 11 人日；如果系统测试没有查出 25%的故障，则节约的成本是 20.5 人日。检查的成本大约是 3.5 人日，但是它产生了那么多节约。这个案例清楚地表明：如果我们在代码检查上多花 1 天时间，则对于该产品，我们能够期望在开发周期后期修复故障时少花 3 到 6 天时间！

实验大约延续了 2 周时间，但是达到了非常好的效果。结果使开发人员和管理人员都信服应当尝试检查方法。实验数据还指出：如果代码简单或者代码较少，则检查的效果就较低（在更少的 SER 中，检查的好处是不明显的）。因此，银行业产品开发团队制定策略将 SER 分成三类（简单、中等复杂和复杂），并考虑对复杂模块执行正式检查。

10.5 小　　结

评审的目的是：由一组对等的评审人员通过一个正式的且结构化的评审过程，识别出一个工作产品中存在的故障和问题。评审是有成本效益的，甚至可以用于不能执行的工作产品。评审是改进质量和提高生产率以及监督项目状态的重要技术。

下面给出了从 Infosys 公司采用的评审方法中吸取的经验教训：

- 评审团队应包括外部专家，以便增进项目团队的才干。
- 采用一个良好定义的结构化的评审过程，具有明确的指南和形式化的数据收集。该过程应当包括规划、自我评审和小组会议。
- 评审期间，重点关注发现故障和问题。而故障和问题的解决是以后的事情。
- 必要时，采用单人对工作产品进行评审。对于单人评审，遵循与小组评审相同的过程和数据收集指南。
- 监督每次评审的有效性。根据以往的数据制定绩效期望，并用它们评价评审的有效性。
- 如果某次评审的绩效没有达到期望的要求，则分析原因并采取纠正和预防措施。
- 为了理解评审的影响，在项目中执行一个简单的实验。对于组织成员来讲，本组织中的数据比外界的任何数据都有更好的说服力。

关于CMM，本节所描述的评审实践满足CMM-3的互相评审KPA。监督和控制方法满足CMM-4的定量化项目管理和软件质量管理KPA的需求。各种工作产品的评审满足很多KPA的评审需求。

10.6 参考文献

[1] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM System Journal*, (3), 1976.

[2] M. E. Fagan. Advances in software inspections. *IEEE Transactions on Software Engineering*, SE-12(7), 1986.

[3] T. Gilb and D. Graham. *Software Inspection*. Addison-Wesley, 1993.

[4] R. B. Grady and T. V. Slack. Key lessons learned in achieving widespread inspection use. *IEEE Software*, July 1994.

[5] E. F. Weller. Lessons learned from three years of inspection data. *IEEE Software*, Sept. 1993.

[6] R. G.. Ebenau and S. H. Strauss. *Software Inspection Process*. McGraw Hill, 1993.

[7] D. P. Freedman and G. M. Weinberg. *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products*. Dorset House, 1990.

[8] D. H. Kitson and S. M. Masters. An analysis of SEI software process assessment results: 1987-1991. *Proceedings of the 15th International Conference on Software Engineering*, 1993.

[9] P. Jalote and M. Haragopal. Overcoming the NAH syndorme for inspection deployment. *Proceedings of the 20th International Conference on Software Engineering*, 1998.

第 11 章　项目监督和控制

一个项目计划，不管准备得多么周详，仍然只是一张纸而已。项目执行期间，项目经理就好比是医生，而项目就好比是病人，医生必须仔细地监视病人的健康状况，并在需要时开出正确剂量的药方（纠正措施）。如果在正确的时间开出了正确剂量的药，则病人就会生存下来。但是，如果不能正确地辨认症状，不能按时开出药方，则可以导致症状更加复杂，甚至可能导致病人死亡。下面两个小案例说明了这一点。

案例 A：希瓦（Shiva）是项目经理，负责为一个大型的合股投资公司开发一个安全交易系统。他的团队由许多专职的但是资浅的技术人员组成，他们基本上没有计算机安全方面的经验。前几个模块的单元测试发现了大量故障——比预想的更多。分析时，希瓦得出如下结论：由于程序难度大和编程人员缺乏经验，正在编写的代码会有更多的故障。因此，他认为在集成和系统测试阶段还会发现更多的故障，除非采取某些措施，否则将会有更多的故障随产品一起被交付。希瓦采取的纠正和预防措施是建立一个独立的测试团队，由一个熟谙项目细节和有着适合测试的性情的人担任测试团队领导。通过与客户讨论，他将系统测试阶段延长了 15 天。最后，尽管交付时间延迟了 15 天，但是交付的系统超过了质量目标，验收测试时发现的故障比预期的要少得多。

案例 B：详细设计阶段结束后，巴拉（Bala）项目的里程碑分析表明，尽管进度没有落后，但是工作量已经超过了 40%。因为进度没有落后，所以没有采取任何纠正措施。但是，在一个月后的另一个里程碑处，分析表明该项目的进度落后了 1 周，而巴拉解释说是由特殊环境造成的。最终开发人员延迟 1 个月才完成项目。验收测试时发现的故障数是巴拉的质量目标的许多倍。最后，他的项目在工作量、进度和质量方面都没有取得成功。事后分析表明：他当时误解了系统的范围，最终低估了工作量。如果在设计里程碑之后第一次出现问题时就重新协商系统范围和进度，最终结果很可能就完全不同了。

这两个小案例引出了两个有关项目监督的关键问题。第 1，项目经理必须监督项目的实际状态，最好的方法是定量地度量关键参数。[1,2]对于项目经理，软件指标主要用来提供这种监督。第 2，监督本身不解决任何问题。项目经理必须正确地解释数据，并且如果他们发现项目没有遵循计划的路线发展，则他们必须

采取正确的措施把它引到正确的路线上来。该数据集提供了有关当前状态的反馈信息和任何所需的纠正措施，构成了项目管理的基本范式。图 11.1 说明了该控制循环。

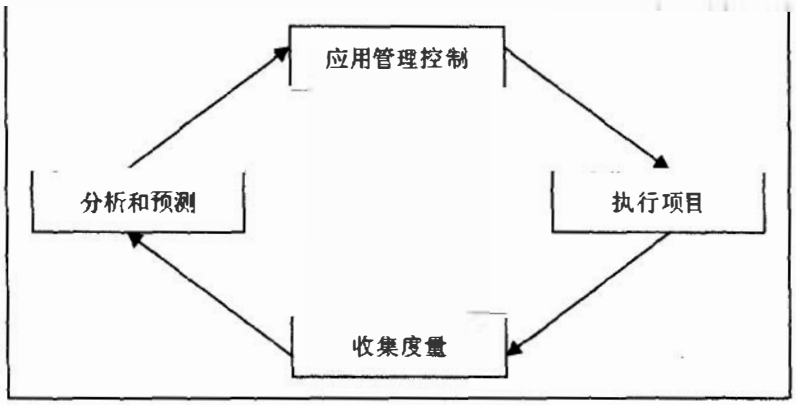


图 11.1　项目监督和控制循环

本章描述 Infosys 公司如何应用监督和控制循环。这是本书最长的一章，讲述了一系列监督任务，包括状态报告、里程碑分析、通过 SPC 的事件级控制、过程审计以及故障预防分析。

11.1　项目跟踪

首先让我们分析一下 Infosys 公司基本的项目跟踪。一旦制定了详细的项目计划（其中列出所有可调度的任务、它们的开始日期和结束日期，以及为它们分配的人员），至少项目经理必须跟踪这些任务的执行。这种努力构成了任务跟踪。此外，项目经理必须跟踪突然出现的未解决的问题和检测到的故障。该跟踪的结果记录在项目的状态报告中。该状态报告还要交给高级管理人员和客户，从而保证他们掌握项目状况。

11.1.1　任务跟踪

项目跟踪的首要任务之一是确保按时完成计划的任务。前文已提过，Infosys 公司通常用 MicrosoftProject（MSP）调度任务。因此，MSP也用于任务跟踪。

每天（频度可以有所变化），项目经理检查安排的任务的状态，然后更新它们在 MSP 中的状态。尽管 MSP 允许用户指定某个任务部分地完成了，但是为了跟踪计，一个任务在完成之前通常规定为 0%完成。任务完成时，标记为 100%完成。这种跟踪层次已经足够了，因为任务已被分解成不过 1 到 2 天就能完成的最底层的、可调度的任务。使用 MSP 功能，项目经理就可以确定哪个更高层的任务落后了，已完成任务的百分比，任务落后对整个项目的执行有什么影响，等等。对于当前阶段（下一个里程碑），基于已完成的任务以及它们实际所花的时间，项目经理根据需要重新安排其余任务，以保证满足里程碑的要求。

11.1.2 故障跟踪

故障跟踪是跟踪的另一个要点。第 7 章已经提到过，Infosys 公司采用一个故障跟踪系统来跟踪故障。一旦关于某个故障的信息被输入到该系统中，直到它被修复之前一直保持未解决状态。当故障排除得到确认之后，故障就被标记为“已解决”。这样，每个故障都被记录下来并在解决之前一直被跟踪。有时，项目经理通过记录每个故障的提交日期和解决日期来跟踪故障的状态。项目收尾时，在理想情况下不应存在未解决的错误（或者作出某种慎重的考虑暂不纠正它们）。在任何时候，项目经理可以检查总的故障引入率和故障解决率，以及有助于检查提交的故障数和解决的故障数之间的差异是否在控制中的信息。

表 11.1 给出了 ACIC 项目案例研究的部分故障数据。此外，还有一些列（表中没有给出）保存提交人 ID、提交日期、所有者 ID（即，负责修改该故障的人的 ID）以及解决日期。从表中可以看出，对于每个故障，记录了诸如引入阶段、检测阶段、严重性等级等标准信息。

11.1.3 问题跟踪

项目执行期间不可避免地会出现很多小事情或者事实澄清，这些就是所谓的问题。问题管理是任何一个项目经理的重要任务，因为这样的问题大量存在并有可能延误一个项目。例如澄清客户需求的真相，在这个问题解决之前，很多任务都会被耽搁。很多这样的任务有可能停止某些任务。因此，项目经理正确地跟踪和管理问题是很重要的。

表 11.1　ACIC 项目的故障数据

ID	故障名称	描述	引入阶段	检测阶段	故障类型	严重性
7	多余的导入语句	在 ActivitiesMaintenanceJPanel 中两次导入 AccountSearchResult 类	编码	代码评审	多余的代码	次要故障
10	没有遵循命名约定	本应名为 ActivitiesMaintenanceJPanel 的面板被命名为 ActivityMaintJPanel	编码	代码评审	标准	装饰性故障
15	事务处理	不需要提交最初用于阅读的事务，因为其他方法可以使用它。这可以在 addTask 和 updateTask 中完成	编码	代码评审	逻辑	次要故障
16	面板结构不正确	在 View History（视图历史）的 Options/Services（选项/服务）面板上，Regy 块面板和 Options/Services 面板放在一个父面板上。然而，正确的方法应当是把它放在 Option Services 面板上	详细设计	详细设计评审	UI	次要故障
21	没有遵循分层法	业务模型 Party 被导入 ViewPartyHistoryPanel	编码	代码评审	体系结构	主要故障
22	没有遵循 UI 标准	屏幕标签不是黑色的	编码	代码评审	标准	装饰性故障
26	TypeCasting 不正确	getIndividualNames 和 getOrganizationNames 方法中 IindividualName 和 IorganizationName 的 TypeCasting 应改成 Iname	编码	代码评审	逻辑	主要故障
27	赋值不正确	作者需要把现有向量的值赋给另一个向量。赋值没有正确地实现，而是建立了一个与现有向量一样大的新向量	编码	代码评审	逻辑	关键性故障
30	接受空值	需要检查是否允许空值	编码	单元测试	逻辑	主要故障
35	财经信息	单击其他面板不应清除屏幕上显示财经信息的字段	编码	单元测试	UI	主要故障

（续表）

ID	故障名称	描述	引入阶段	检测阶段	故障类型	严重性
39	财经信息	没有完成 ApplicationContext 面板刷新的异常处理	编码	单元测试	UI	主要故障
152	标签设置	面板没有设置标签移动	编码	集成测试	绩效	装饰性故障
153	禁用的添加按钮	在账户角色表（Account Roles Table）中选定一个角色之后，Address 中的添加按钮被禁用	编码	集成测试	绩效	主要故障
154	清除按钮没有清除 AssociateAccounts 表	地址维护面板中的清除按钮没有消除关联的账户表	编码	集成测试	绩效	次要故障
155	电话分离没有起作用	在电话维护面板中，电话分离没有起作用	编码	集成测试	绩效	主要故障
156	电话更新没有起作用	更新一个电话号码时，没有保存变更	编码	集成测试	绩效	主要故障
157	E-mail 更新没有起作用	遵循某个动作序列——不用单击更新按钮的情况下更新一个 E-mail，选择另一个角色，单击确认对话框上的 Yes 按钮——然后，选择前一个角色，则面板只显示已经更新的 E-mail，而不显示其他 E-mail	编码	集成测试	绩效	主要故障
158	没有更新账户类型	当变更账户类型并单击更新时，账户类型没有被保存	编码	集成测试	绩效	主要故障

一种方法是仅仅记录问题并在适当的时候检查它们。但是，因为任何团队成员都会出现问题，所以问题表会变得很长。因此，用正式的方法跟踪它们比较有用。因此，项目通常建立一个问题日志。

在该问题日志中，问题出现时就记录它们以及它们的相关信息。当问题解决时，就把它们标记为已解决。对于该问题日志，项目经理可以采用电子表格、维护一个文档、使用一个问题跟踪器实用程序或者使用故障跟踪系统。如果采用了一种自动跟踪工具，则简单查询成为可能。表 11.2 给出了 ACIC 项目的部分问题日志，使你了解问题日志中保存的项目类型（实际的日志包含提交日期和结束日期以及其他跟踪信息）。

表 11.2　ACIC 项目的问题日志示例

问题描述	注解/问题解决注解	状态
记录的哪些字段将被发送到 ACIC的业务伙伴?	已更新的用例	已解决
UC3 中的候选情节不明确。为什么 9 表示人工更新?	用例 4 增加的候选情节	已解决
向导只允许 Stnts 和 Confirms 中的一个值。为什么它应与维护屏幕不同呢?	用例中增加说明	未解决
当地址变更应当被发送到 ACIC 业务伙伴时，澄清所有条件		未解决
Synergy 中的主要指示器被取消了。如果必须恢复它，则必须在用例中作为一个需求提到它	参考前一天发送的 PPT。屏幕上有此字段	已解决
更新时，没有讨论候选情节。如果更新的 e-mail 与其他账户有关该怎么办呢?	用例中增加说明	已解决
在显示服务选项的历史的表中，所有三个选项类型都有一个时间戳。这在分别处理不同选项时不会出错吗?		未解决
至于表填充，应当向表中填入某些有意义的数据		未解决
如果对应赋税的角色或者感兴趣的团体作了变更，它应影响其他账户吗?	在用例中提到这个问题。请查看主要情节中的“事件流”	已解决
国家组合框需要用国家代码值预先填入吗，还是必须把它转变成一个文本字段?		未解决
可以删除文本字段 ForeignAddressLine4 吗?		未解决
维护团队的外国地址和外国电话号码还没有作好集成准备		未解决
VAJ 3.0 变换将影响该项目	只有在项目完成后才执行此变换	已解决

项目经理必须定期检查问题的状态，尤其在所有团队成员都可以输入它的情况下。第 8 章已经提到过，未解决的问题会造成项目的风险。如果问题在很长时间内都没有得到解决，则必须采用项目管理计划中规定的提交渠道来解决它们。

11.1.4　状态报告

状态报告是定期向高级管理人员和客户通报项目状态的主要机制。项目管理计划应规定接收状态报告的团体。通常，状态报告每周制定一份，包含如下条目：

- 客户投诉
- 本周实现的里程碑
- 本周没有实现的里程碑及其原因
- 下周计划的里程碑

- 需要说明或者注意的问题
- 如果需要，提交问题
- 按里程碑估计的工作与可用时间

显然，状态报告主要是为了保证项目继续按计划进展并解决未决的问题。项目经理也可以检查项目的“舒适水平（comfort level）”，查看可用的时间是否满足所需的工作量。项目经理还可以增加更多的条目，诸如需求变更量，并可以删除那些已显示条目中的一些条目。

图 11.2 给出了一个 ACIC 项目的状态报告示例。这个周状态报告主要关心上一周完成了什么工作、需要完成什么工作、哪些问题还没有解决，等等。

状态总结

项目	生命期阶段	下一个里程碑日期	完成百分比	资源量	备注
2.0 版	构建阶段	2000 年 7 月 7 日	90%	4	
3.0 版	第 1 次细化 第 2 次细化	2000 年 8 月 28 日	第 1 次细化：95% 第 2 次细化：20%	5.5	有 2 个细化阶段和 3 个构造阶段

已完成的任务和未完成的任务

	已完成的任务	未完成的任务
2.0 版	● 第 2 次迭代的构造已经完成 ● 代码评审已经完成 ● 第 2 和 3 次迭代的测试计划已经完成 ● 已建立了独立的测试环境	没有
3.0 版	● 作为第 1 次细化的一部分，设计用了 10 个用例，而实现用了 4 个关键用例。该任务完成了 95% ● 第 2 次迭代开始	需求评审将在下周开始

下周（从 2000 年 7 月 4 日到 2000 年 7 月 9 日）计划的任务

2.0 版	● 完成第 3 次迭代的构造阶段 ● 继续进行第 1 次迭代的独立测试 ● 执行第 2 和 3 次迭代的代码评审 ● 完成第 2 和 3 次迭代的测试计划
3.0 版	● 第 2 次迭代 ● 完成体系结构文档 ● 评审 BAR ● 安排 PMR 评审时间

问题/其他

未解决的问题　　没有
其他　　没有

图 11.2　ACIC 项目状态报告

11.2　里程碑分析

状态报告提供了定期监督项目的机制。它主要解决进度有没有得到满足。状态报告的一个主要优点是:它们不需要太多的指标数据或者分析。本节讨论 Infosys 公司如何用指标数据来评价项目在各里程碑处的状态。

如果项目计划是经仔细推导而得到的并且被评价过，则在遵循计划的情况下项目可能会成功。因此，里程碑分析应当将项目计划和进度计划作为基准，并将计划的进展与实际的进展进行比较。两种通用的基于指标的跟踪方法采用了该策略：成本-进度-里程碑图和挣值分析方法。[5]分析计划的进展与实际进展被认为是项目管理的最佳实践。[3]在 Infosys 公司，在里程碑处还要跟踪计划的进展和实际进展。

因为监督和控制的指标分析的主要目标是及时地采取纠正和预防措施，所以应当定期进行这种分析。为此，项目经理可以在客户规定的里程碑之上定义里程碑，以便相邻的里程碑间只有几周之隔。

11.2.1　实际的与计划的工作量和进度分析

对于进度和工作量，一个项目的实际进展很可能与计划有偏差。然而，工作量或者进度的少量偏差可以被认为是“正常的”，并且不值得特别注意。另一方面，“重大”偏差意味着项目可能导致失败，因此需要进一步的分析和控制措施。

为了区别正常偏差与重大偏差，项目计划应设定可接受的偏差极限，这在第 7 章已讨论过。Infosys 公司的很多项目的工作量偏差的极限大约是 20%，而进度偏差大约是 10%。对于一个项目的实际的偏差极限在管理计划中规定。

如果里程碑处的偏差超过这些极限，则可能隐含着项目将会遇到麻烦，并且可能不能满足它的目标；如果时间紧迫，项目成员很可能会采取不符合要求的捷径。这种情况要求项目经理理解偏差的原因，并在必要时采取纠正和预防措施。表 11.3 列出了分析指南和项目经理可以考虑采用的控制措施。

表 11.3 给出了两种不同情况下的建议：估计的工作量太低或者估计的工作量太高。此外还给出了一些原因，以及可以采取的措施。例如，如果估计太低，可能的原因是估计太乐观、资源利用率低（即，项目存在资源浪费）、缺乏一种关键

资源，或者项目成员的经验不够丰富。对于每种原因，都建议可以采用的某些控制措施。例如，如果估计太乐观，则项目经理可以修改估计、申请资源或者尽力缩小项目规模。

表 11.3 工作量/进度绩效指南

可能的原因	考虑采取的措施
如果实际工作量少于估计工作量，并且两者之间的偏差超出了允许的极限	
对程序的工作量估计得太高，或者项目团队比期望的有更多的领域知识或者更丰富的经验	● 重新估计未来模块的工作量 ● 释放资源
至今还没有彻底地执行任务	● 至今完成的评审任务和工作产品的进度评审没有被审查过 ● 检查问题日志
如果实际工作量高于估计工作量，并且两者之间的偏差超出了允许的极限	
领域知识不够	● 安排培训时间
作者的软件/编码经验不够	● 重新分配任务，以充分发挥作者现有的经验
新技术领域	● 重新估计或者申请资源 ● 协商交付日期
估计太乐观	● 标识出需要额外工作量的主要组件，并修改未来任务的估计 ● 申请资源 ● 通过协商以缩小项目目标的规模
资源优化程度低	● 重新安排任务的时间，并重新分配各任务的优先级，并标识和消除“等待时间”
缺乏某种关键资源	● 提交该问题 ● 获取后备资源 ● 重新制定进度计划，牢记关键资源
由于早期阶段的结果质量太低，导致太多的重复工作	● 标识出问题的来源并改正它们 ● 变更项目的进度计划

工作量偏差的模式也是有助于分析的。如果从一个里程碑到另一个里程碑的工作量偏差一致地上升，则即使偏差在极限范围内，也必须保证采取某些措施。同样，如果采取了控制措施，并且在下一个里程碑处的偏差百分比下降了，则表明所采取的措施起到了希望的效果。另一方面，如果在采取措施后工作量偏差变得更大，则意味着所采取的措施没有很好地发挥作用，并且可能需要采取更加强有力的措施。

除了以往的绩效外，项目经理可以在里程碑处检查项目的其余任务的预兆。对于工作量，在里程碑分析时既要报告可用的资源和所需的资源。如果有效的总

工作量（基于资源量）严重低于所需的工作量，则显然必须采取某种措施。实现这种监督以便及时采取措施是这种分析的目标。

11.2.2　质量监督

为了监督第三维——质量——所采用的主要指标是发现的故障数。此外，还要监督评审的次数和状态及故障预防措施的效果。因为规划也对故障数进行了估计，所以要同进度和工作量一样对故障进行评价。即，使用项目计划预测的各阶段故障数和实际发现的故障数，项目经理准备实际故障数与估计故障数的分析。如果偏差超出项目所设的阈值，则必须根据项目经理作出的使项目在控制之下的决策，分析偏差的原因。表 11.4 给出了与可能的原因和可能的控制措施有关的指南。（第 10 章描述了评价评审效果的指南。）

表 11.4　实际发现的故障数不同于估计的故障数时的指南

可能的原因	应考虑的措施
如果实际的故障数低于估计的故障数	
工作产品质量很高	● 标识原因，并查明项目或者过程是否有什么可吸取的经验
测试不充分	● 检查测试所花的工作量；评审测试计划并增强它 ● 安排再次测试的时间
非常彻底地执行了早期的质量控制任务	● 检查项目的所有评审和测试记录 ● 检查项目或者过程是否有什么可吸取的经验
故障估计太高	● 标识原因，并纠正估计
如果实际的故障数高于估计的故障数	
至今为止所执行的质量控制任务不够充分	● 检查所有测试和评审记录 ● 在继续测试之前安排关键模块的评审
计划的评审和单元测试不充分	● 增强测试计划和安排再次测试 ● 评审估计和验收计划
故障估计太低	● 确定原因，并纠正估计

对于测试和评审的质量控制任务，除了发现的故障数，项目经理还要分析这些任务实际所花的工作量和估计的工作量。当实际发现的故障数显著地不同于估计的故障数时，这些数据有助于分析当时的情况。例如，如果系统测试后发现的故障太少，并且系统测试所花的总时间也很少，则测试结果的原因就会显得更为明显。

要监督的第 2 个指标是执行的评审次数。由于时间紧迫，很容易跳过计划的评审。因为这些评审是为了实现项目的质量目标而开发，是质量计划的一个重要组成部分，所以它们的圆满执行是很重要的。因此，里程碑报告指明计划了哪些评审，以及哪些评审实际被执行了。在每次评审完成时，还要监督评审的实际绩效，这在第 10 章讨论过。

11.2.3 与风险有关的监督

在里程碑处还要对风险及其相关的任务进行监督。第 7 章已经讨论过，项目的风险不是固定不变；风险认识随时间而变化，也随所执行的风险缓和措施而变化。因此，观察风险并记录至今为止所采取的风险缓和措施的效果，这样做是重要的。为此，里程碑分析报告应报告当前的风险感以及当前的风险缓和措施的状态。显然，如果某种风险的风险暴露度还没有减少，则意味着风险缓和措施没有起到期望的效果。因此，必须对风险及其缓和措施进行评估。

一种重要的风险缓和措施是培训，通过培训可以解决很多种风险。培训也是任何牵涉到新技术或者有新手参加的项目的一个重要成份，如果不进行正确的培训，则会引入新的风险。为了防止培训监督因失去控制而落空，里程碑报告应描述计划的与项目有关的培训和实际执行的培训。

需求变更也会引入项目的风险，因为它们对成本、进度和质量有负面影响。第 3 章说明了管理需求变更的过程。里程碑分析时，要报告已经被申请的变更及其对工作量和进度的影响。

如果问题长时间没有得到解决，通常也会转变成风险。因此，里程碑分析还要报告问题的状态。此外，还要报告客户投诉，因为客户投诉明确地表明了项目的严重风险。尤其是，新接到的客户投诉量、已解决的客户投诉量以及未决的客户投诉量都要进行报告。

11.2.4 ACIC 项目的里程碑分析

让我们看看一个关于 ACIC 项目的里程碑分析报告（其他例子参见我以前写的书）。[6]如图 11.3 所示，在 ACIC 项目的第一次构造迭代完成时执行里程碑分析。分析表明进度和工作量基本上没有落后，并且在第一次构造迭代完成之前可用的工作量接近所需的工作量。因此，项目在这些方面得到了控制，不需要采取

任何措施。然而，在质量方面，报告指明故障水平比期望的更高，因为引入了更多的故障。因此，要采取某种故障预防措施。此外，项目其余任务的故障引入率也进行了修改。

里程碑名称：构造阶段：第 1 次迭代　　日期：2000 年 8 月 18 日

进度（里程碑）

计划的日期	实际日期	落后时间
2000 年 7 月 21 日	2000 年 7 月 21 日	0

偏差原因：N/A
使进度恢复控制应采取的措施：无
对项目的总体影响：无

工作量（人时）

该里程碑前估计的工作量	该里程碑前的实际工作量	偏差	从该里程碑到项目收尾的估计的工作量	项目收尾前可用的工作量
2019	2191	9	2231	2059

引起偏差的原因：N/A
采取的措施：无
对项目的总体影响：无

故障

编号	QC 任务	工作产品的规模	MSP估计的工作量（人时）	实际工作量（人时）	估计的故障数	实际故障数	故障数偏差
1	代码评审和单元测试	大约4000 行	31	25.5	25	57	125%

出现偏差的原因：其中一个用例有太多的故障。
采取的措施：已经采取了很多措施来防止这些故障发生。在因果分析报告中给出了更加详细的说明。
对质量目标的影响：项目的故障引入率可能比计划的更高。故障引入率调整了 20%，并且质量计划也作了相应地修改。但是，项目目标保持不变。为了使未来的影响最小，应当更加强有力地执行 DP（故障预防）任务。
故障预防任务
一种因果分析在根源分析后进行，然后根据这种分析建议某些预防措施。这些是将要被执行的措施。这些预防措施的潜在影响及其分析在因果分析报告中给出。
需求变更跟踪

迄今为止的主要需求变更的总数	0
迄今为止的次要需求变更的总数	0

图 11.3　ACIC 项目的里程碑分析报告

风险

	以前的状态					当前状态		
序号	概率	影响	暴露度	风险条目	风险缓和措施的状态	概率	影响	暴露度
1	0.9	9	8.1	VAJ 3.0 转换日期不知道，如果计划在3.0版交付产品，则可能会影响该项目	已提交给客户，他们已注意这个问题	0.9	9	8.1
2	0.9	9	8.1	屏幕大小调整的日期不知道	工作量分级后交给客户。还没有客户的答复	0.9	9	8.1
3	0.8	8	6.4	3.0版依赖于联机维护团队的某些组件	问题由维护团队提出。他们没有从事此工作的时间	0.9	8	7.2
4	0.3	7	2.1	3.0 版使用的某些组件依赖于2.0版	2.0版有望按时完成	0.2	7	1.4
5	0.9	1	0.9	必须执行一些主机工作，并且该工作没有制定计划	客户已经批准了一个主机资源。不再是一个风险	0	0	0

客户投诉

提出的客户投诉数量	0
已解决的客户投诉数量	0
未解决的客户投诉数量	0

培训

计划的培训	为 Bhaskar 和 Balajee 设计 J2EE分布式应用
实际的培训	完成了计划的培训

小组评审

计划的小组评审	没有
实际举行的小组评审	没有

问题：风险 1，2，3 仍然有问题。随后与客户方一起提出解决这些问题的方案。

图 11.3 ACIC 项目的里程碑分析报告（续）

项目经理重新评估了风险，并发现：尽管项目目前面临的风险不同于项目开始时的风险，但是自上一个里程碑以来，风险及其危险等级基本没有改变。分析报告还指明已经执行了计划的培训任务。

11.3　用 SPC（统计过程控制）进行任务级分析

里程碑分析用指标来监督在定义的里程碑处的项目状态，并在需要时建议纠正措施。然而，采用基于里程碑的分析，很难确定自上一个里程碑以来执行的那么多任务，其中哪一个任务可能是引起绩效下降的原因。这种情况类似于执行系统测试：当系统由很多单元构成时，调试变得更加困难。单元测试期间的调试非常容易。因为绩效差的原因很难查明，所以限制了可以采用的纠正措施。

为了提供更细的控制，在 Infosys 公司还要执行任务级分析。进行任务级分析时，在一个任务被执行以后立即对某些指标进行分析。该分析用来评价任务绩效的有效性。如果任务的执行令人不满意，可以及时采取行动纠正它并防止重复发生。

任务级分析通常用统计过程控制（SPC）来完成。在 Infosys 公司，评审和单元测试是两个可以应用 SPC 的任务。使用控制图，项目经理可以直接评价评审或者单元测试的效果。如果绩效不好，可以采取必要的措施。

注意，设计、详细设计、系统测试等众多任务都已经通过里程碑分析被单独评价过了，因为它们通常在里程碑处开始和结束。因此，任务级监督集中解决与编码有关的评审和单元测试任务。

第 7 章讨论了统计过程控制的基础，第 10 章说明了用 SPC 监督和控制评审。这里我们讨论单元测试的任务级监督和控制——另一种应用 SPC 的任务。单元测试方法类似于评审监督所用的方法。被监督的主要绩效特征是单元测试时检测到的故障密度。根据单元测试的以往数据，就可以求得控制极限（即，可接受的故障密度范围）。在每次单元测试结束时，根据该范围检查故障密度。如果故障密度落在该范围外，则必须采取纠正或者预防措施。前面为单元测试提供的指南用来确定哪些措施是必须的（分析时，项目经理可以使用支持信息，诸如任务所花的工作量、以前的任务所花的工作量以及所发现的故障）。图 11.4 表示某个编程语言的单元测试时故障密度控制图。故障密度可以根据每 LOC 的故障数或者每人时的故障数在图上画出。表 11.5 表示单元测试的能力基准，给出了几种语言的故障率。

至于评审，项目经理可以使用 SPC 工具来实现单元测试的任务控制。该工具包含相关的以往绩效数据，诸如故障引入率、故障排除百分比，等等。使用该数据，它就能确定控制极限以及期望的极限。当单元测试的绩效数据输入 SPC 工具

时，工具立即可以告知绩效有没有在极限外。如果在极限外，则进行进一步分析以确定在需要时采取什么措施。

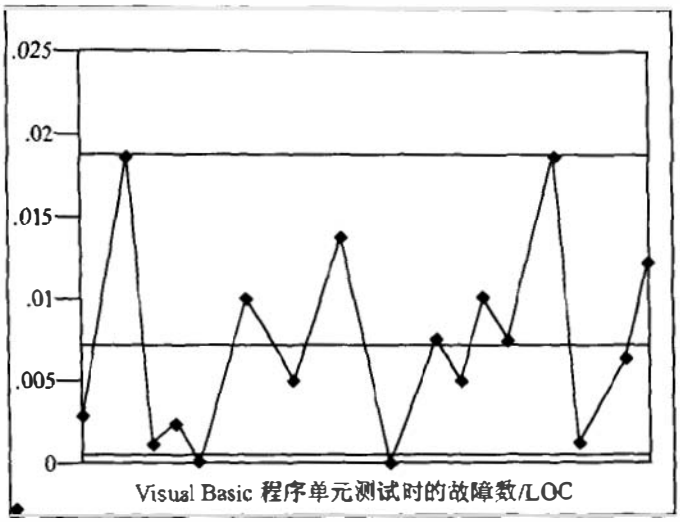


图 11.4 Visual Basic 程序单元测试的控制图

表 11.5 几种语言的单元测试故障率

语言	故障率范围（平均故障率）
PB	0.0003-0.0266 个故障/LOC（平均：0.008）
C	0.0004-0.0206 个故障/LOC（平均：0.0052）
C++	0.00009-0.0067 个故障/LOC（平均：0.0017）
RPG	0.0006-0.0075 个故障/LOC（平均：0.0025）

11.4 故障分析和预防

故障预防旨在学习至今项目中所发现的故障，以预防项目的其余任务出现故障。在第 5 章已经讨论过，通常在一个项目中执行两次故障预防任务：一次是已经编码了 20%模块并且经过了单元测试，另一次是已经编码了 50%的模块并经过了单元测试。故障预防的主要任务是执行帕累多（Pareto）分析以识别出主要故障类型，执行因果分析以识别出故障的原因，以及确定解决原因的方案。本节讨论如何在一个项目中执行这些任务。

11.4.1　执行帕累多分析

帕累多分析是一种常见的用于原因分析的统计技术，是质量管理的主要工具之一。[7,8]有时也把它称为 80-20 规则：即 80%的问题可能来自 20%的原因。在软件方面，意味着 80%的故障可能来自 20%的因素，或者说可以在 20%的代码中发现 80%的故障。

故障预防的第一步是根据故障数据绘出一幅帕累多图。根据故障数据计算出所发现的不同类型的故障数，并按降序绘成一幅柱状统计图表。连同柱状统计图表一起，还要在这同一副图上绘出另一个图表，在 X 轴上从左到右显示各种故障类型的累积故障数。帕累多图既可以从视觉上也可以从数量上明确而直观地表明哪些是主要故障类型，以及哪些故障加在一起构成了 80%-85%的总故障。除了绘出故障数外，通过给不同类型的故障分配不同的权值，也可以绘出不同类型故障的加权和。

执行帕累多分析的总体方法如下：

（1）列出至今已发现的所有故障。
（2）计算各种故障类型的故障数。
（3）根据故障类型，按故障数降序排列故障。
（4）计算每种故障类型占检测到的总故障数的百分比。
（5）标识出占 80%的总故障根源的故障类型。

例如，考虑一下图 11.5 所示的 ACIC 项目故障数据的帕累多图。该项目是向一个现有的系统添加功能。关于以往功能增强的所有故障数据都用于此分析。从图中可以看到，故障数最高的是逻辑故障，接着是用户接口故障和标准故障。这三种故障加在一起占总故障的 88%以上，而最高的两种故障占总故障数的 75%以上。显然，故障预防的目标应当是前面 2 种或者 3 种故障类型。

11.4.2　执行因果分析

帕累多图有助于确定项目中至今已发现的主要故障类型，如果不采取措施，还有可能在项目的其余任务中发现这些故障。你要尽量使这些故障对以后的“影响”最小。为了减少这些故障，你必须找到它们的主要原因，然后尽力消除它们。因一果（cause-effect，简称 CE）图可用于确定观测到的影响的原因。[7,8]例如，

因一果图可用于确定ACE项目中GUI故障（或者逻辑故障）数量高的主要原因。CE图的主要目的是图形化地表示影响及其各种可能的原因之间的关系。理解原因有助于确定消除它们的解决方案。

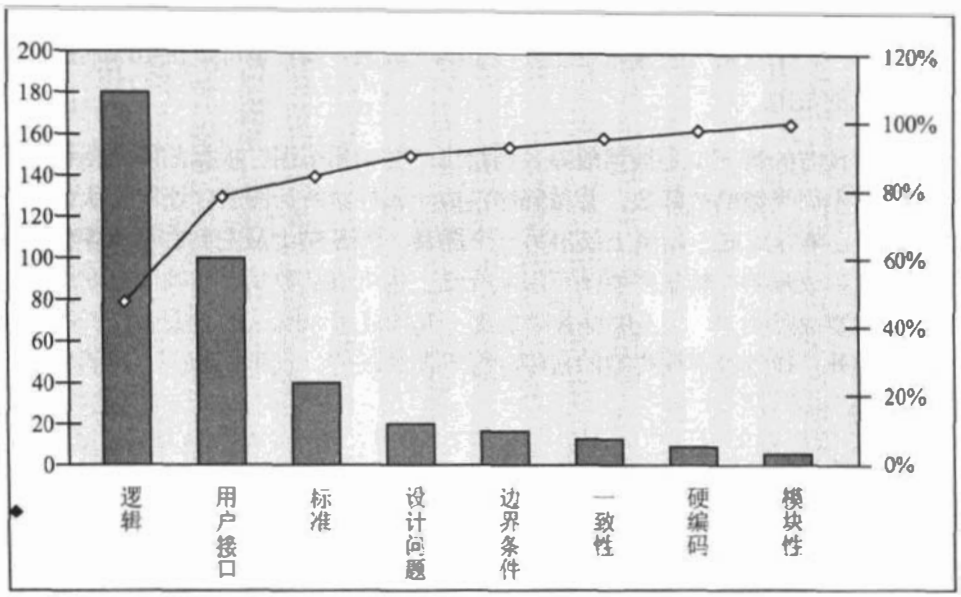


图11.5 ACIC项目所发现的故障的帕累多图

建立一幅因一果图的第一步是确定要进行的影响分析。在ACE例子中，影响可能是“太多的GUI错误”。为了确定该影响的原因，首先确立某几类主要的原因。对于制造业，这些主要的原因通常是人力、机器、方法、材料、度量和环境。对于Infosys公司采用的因果分析，主要故障原因的标准集是过程、人员、技术和培训（培训之所以同人员分开是因为它经常出现）。该图的主要结构是将影响作为一个方框显示在右边；然后自该方框画出一条水平直线，每个主要原因通过一根箭头线与该主线相连接。

为了分析原因，关键是对每个主要原因提出如下问题“为什么该原因会产生这种影响？”这些问题的答案就成了分原因，可以用短的水平线表示它们，并与表示主要原因的直线相连接。然后，对识别出的原因提出同样的问题。这种“为什么-为什么-为什么”过程一直重复下去，直到已经确定了所有根源——即，对原因问“为什么”不再有任何意义。当在图中标出了所有的原因后，最后的图看起来象一种鱼骨形结构，因此这种因一果图也称为鱼骨（fish-bone）图，或者用

它的发明者的名字命名为石川（Ishikawa）图。

绘制一幅因一果图的主要步骤如下：[8]

（1）明确定义要研究的问题（影响）。对于故障预防，问题通常是“太多的 X 类型故障”。

（2）从左到右画一根箭头线，并在头部画一个包含结果的方框。这条水平直线该图的主干。

（3）确定主要的原因类别。这些分类既可以是标准分类，也可以对标准分类作出某些变化，以适应问题。

（4）将这些主要的原因类别写在方框中，并用三角箭头将它们与主干相连接。这些故障种类形成了因一果图的骨架。

（5）通过对每个主要的原因反复地提问“为什么该主要原因产生这样的影响？”集体讨论主要原因的分原因。

（6）将这些分原因添加到图中，并聚合在主要原因的骨架周围。如果需要，进一步细分这些原因。如果发现对上述问题的回答已经不再有任何意义，则停止原因细分。

绘出鱼骨图后，就标识出了正在研究的影响的所有原因。但是，初始鱼骨图很可能有太多的原因。显然，其中一些原因比其他原因有更大的影响。因此，在根源分析之前，主要通过讨论确定几种主要原因。对于故障预防，可以对帕累多分析发现的最主要的 1 种或者 2 种故障执行整个这样的任务。

图 11.6 表示 ACE 项目的鱼骨图。在这次分析中，在一次集体讨论会上讨论了三种主要故障的原因。因此，我们的影响是“太多的逻辑/GUI/标准故障”。当我们提问“为什么人们会产生那么多的逻辑或者 GUI或者标准故障呢？”时，我们就确定了其中某些（比较明显的）原因：缺乏培训、疏忽（即，不够重视），缺乏专门技术。同样，当我们提问“为什么过程会产生这么多的逻辑/GUI/标准故障呢？”时，原因可能是“没有综合记录标准”和“人们没有注意标准”。技术原因可能是“不明确的规范”和“工具的技术问题”。因果分析的集体讨论会将产生更多的原因。列出会议上提出的所有建议后，故障预防团队通过研究每个故障并确定其原因，划定故障的危险等级。那些最频繁出现的原因具有高的危险等级。如图 11.6所示。

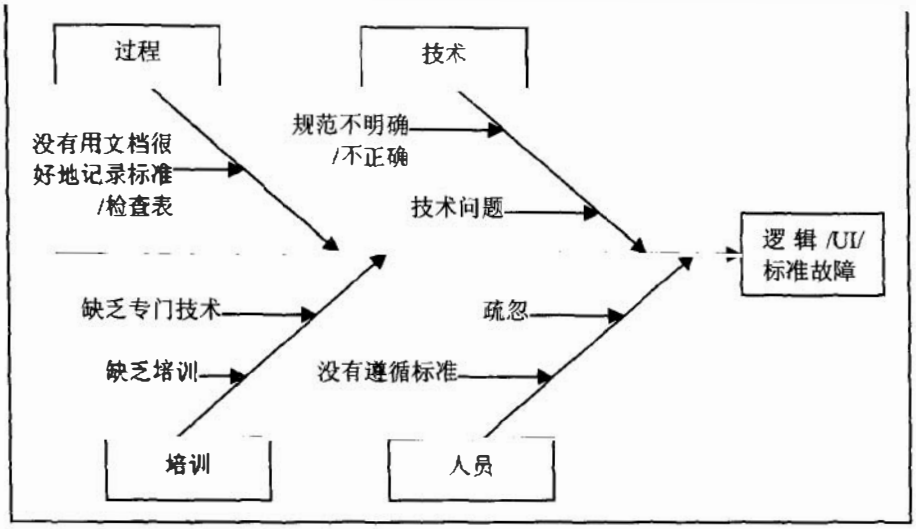


图 11.6 ACE 项目的因一果图

11.4.3 制定和实施解决方案

到此为止，我们已经讨论了如何确定经常发生的故障类型以及它们的根源。接下去的工作是采取措施减少故障发生。

基本范式是遵循古训“预防胜于治疗”。对于故障预防，你不必设法“治疗”软件的故障；相反，你要采取预防措施，以便软件不会因故障而“生病”。通用的预防措施是建立或者改进检查表、执行培训程序和评审，以及使用一种专门的工具。当然，有时你必须采取强有力的措施，诸如变更过程或者技术。

像因一果分析那样，通过集体讨论会制定解决方案。因此，因一果分析和制定解决方案通常在同一个会上进行。这也是 Infosys 公司采取的方式。

预防方案是某人必须执行的行动细目。因此，方案的实施是关键。除非方案被实现了，否则它们是无用的。在 Infosys 公司，连同方案一起，还要规定负责实现它的人。然后，这些行动细目就被添加到项目的详细任务进度计划中，并同其他任务一样跟踪它们的实现。表 11.6 表示了 ACE 项目的根源及其预防措施，建议的预防措施是显然的。在项目的 MSP 进度计划中安排它们的时间。

表 11.6　ACE 项目的根源及其建议的解决方案

根源	预防措施	负责人	实现日期
没有遵循标准	● 小组阅读标准（在标准被更新后） ● 保证在执行同类项目时遵循该标准	所有人	15/12/00
没有用文档很好地记录标准/检查表	● 与外界专家一起对标准进行小组评审，然后更新标准	Xxxx	下一周
疏忽（即，不够重视）	● 有效的自我评审 ● 严格的代码评审	所有人	立即执行 立即执行
规范不明确/不正确	● 规范评审	所有人	立即执行
缺乏培训	● 每个新手将执行一个模拟项目，然后对他们的代码进行全面评审和测试 ● 为该模拟项目制定详细规范和测试计划	xxxx	29/12/00
技术问题	● 使人员注意工具的问题，使他们知道如何避免它们 ● 编写有关此问题的 BOK，并使之可用		
缺乏专门技术	● 用文档记录有关谢里丹网格（Sheridan grid）、记录集、活动报告（Active Report）等主题的 BOK	xxxx	31/01/01

实施这些方案的一项重要内容是：查看它们是否有期望的减少故障引入的效果，从而减少返工的工作量。对方案实施后发现的问题进行深入分析，这可以加深对问题的了解。通常，故障预防的后续分析就是为此目的。除了跟踪影响外，这种后续分析具有重大的巩固作用。无论什么都比不上让人们看到这种效益更能说服他们了。因此，除了方案实施外，还应当分析实施的影响。

11.4.4　ACIC 项目的故障预防（DP）

现在，让我们看看 ACIC 案例研究的 DP 过程。第 1 次构造迭代后的故障数据已经被分析过，并且各种故障类型的频度如表 11.7 所示。图 11.7 表明了故障数据的帕累多图。

故障预防任务的主要目的是减少故障引入率。在第 1 次迭代时，ACIC 项目经理知道至少引入了 57 个故障。他根据工作量数据计算了构建阶段的故障引入率，结果为 0.33 个故障/小时。按照计划，应该有大约 70%的故障在构建任务阶段引入，估计的工作量是 110 天（不包括估计的返工工作量）。即，按照质量和工作量计划，编码阶段的故障引入率应该在 0.1 个故障/小时左右。但是在第 1 次迭代后，故障引入率是该数字的 3 倍！显然，需要采取预防措施才能实现该目标。

表 11.7 ACIC 项目第 1 次迭代后的故障数据总结

故障类型	故障数
逻辑故障	19
标准故障	17
冗余代码	11
UI	8
体系结构	2
总计	57

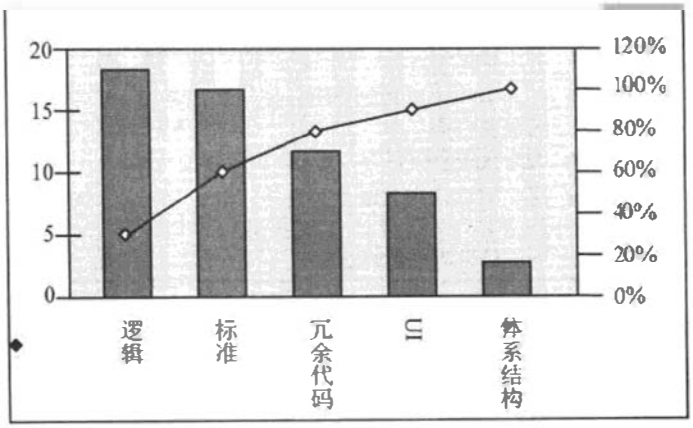


图 11.7 故障的帕累多图

为了显著地减少故障引入率，项目经理决定处理最主要的 3 种故障类型：逻辑、标准和冗余代码。并召开了一次集体讨论会，以确定故障的根源和可能的预防措施。集体讨论的正规程序如下。首先，列出每个人建议的所有可能的原因，然后分离出作为主要事故原因的原因。针对这些原因，讨论可能采取的预防措施，并最终在预防措施上取得一致意见。表 11.8 表示了因果分析会议的最终结论——即，主要根源和要实施的预防措施。其中的很多措施转变成了可调度的任务，并被添加到项目进度计划中，然后在以后执行它（不对那些分配给“自身”的任务进行正式监督）。

表中给定的预防措施都是团队成员的建议；项目经理必须确定它们给出了期望的结果。这些措施在减少故障引入率方面是否成功只能通过故障数据进行检查。

表 11.8　ACIC 项目各种故障类型的根源和预防措施

故障类型（故障数）	根源	预防措施	负责人
标准（17）	缺乏编程经验	培训	自我
	疏忽	开发人员必须仔细阅读编码标准并严格遵循它们	自我
	缺乏对程序规范的使用和需求的理解	(1) 提出一种用 Rational Rose 产生程序规范的方法。(2) 为评审程序规范准备检查表。(3) 为编写程序规范准备指南	XXXX
	没有更新编码标准	更新代码标准，并准备一个文档，列出现有项目特有的 UI 标准	XXXX
冗余代码（11）	缺乏对语言的理解	培训	XXXX
	缺乏对对象模型和数据库的理解	（1）数据库结构培训 （2）开发人员应当全面检查对象模型	由 xxxx 主持 DB 会议
	缺乏对现有代码的理解	在会议上分组讨论，最后确定通用的方法调用集，并标识应当从哪里调用它们	团队
	缺乏对表模型的理解	理解表模型的功能和对表选择的依赖，并通知团队	Xxxx
逻辑（19）	缺乏对现有代码的理解	安排时间阅读代码	自我
	缺乏编程经验	培训	XXXX
	缺乏对顺序图表示的理解	提供 Rational Rose 培训	XXXX
	缺乏对数据库及其有关过程的理解	与前一个故障的预防措施相同	XXXX
	缺乏对对象模型的理解	与前一个故障的预防措施相同	自我
	疏忽	程序员应当更加全面地执行自我测试。举行一个如何测试小部分代码的会议	由 XXXX 培训
	缺乏对用例的理解	开发人员应当检查需求	自我
	缺乏对业务规则的理解	（1）开发人员参考有效的标准来处理各种规则。（2）开发人员评审以往应用的用例，以更好地理解业务规则	自我
	缺乏对故障的理解	故障所有者应当执行评审人员提出的后续措施，应尝试减少任何现有的沟通差距，更加频繁地执行团队/成员关心的问题的后续措施	团队

在第 1 次构造迭代完成后执行故障预防任务。因为 ACIC 项目包括 3 次迭代，因此还要计算后 2 次迭代后的故障引入率。图 11.8 表示了后 2 次迭代后执行的分析的结果。该图明确表示了实施预防措施对故障引入率的影响：引入率从大于 0.33 减少到少于0.1！

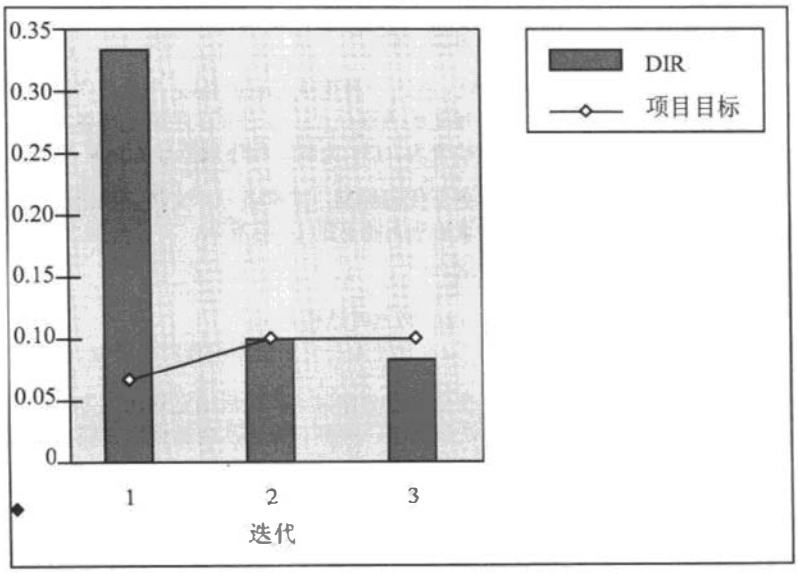


图 11.8 ACIC 项目不同迭代阶段的故障 DIR（引入率）

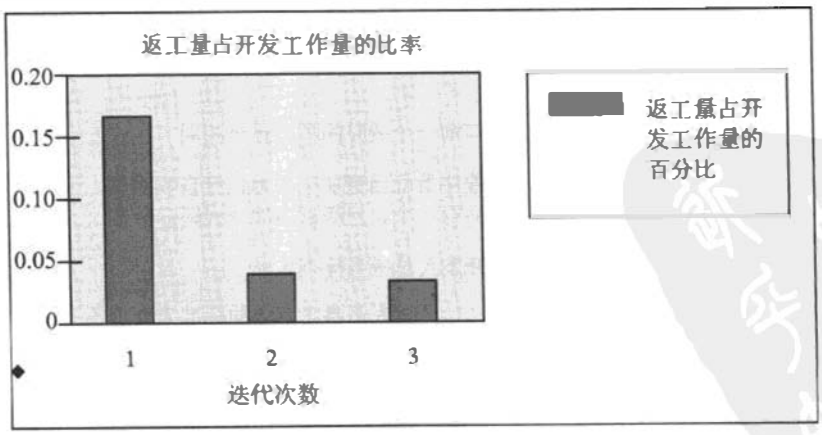


图 11.9 ACIC 项目由于故障预防而减少的返工量

故障引入的减少隐含着要被检测和修复的故障数更少了。因此，一种成功的故障预防措施应当减少测试后的返工工作量。图 11.9 表示了 3 次迭代的返工工作量。该返工工作量根据 WAR（任务周报）求得，因为有不同的代码要返工，此外还要规定程序和模块。第 1 次构造迭代后的返工工作量大约是那次迭代的总工作量的 16%。在后 2 次迭代中该工作量减少为 5%和 3%。因果分析所花的工作量是几小时的数据分析和有关故障预防的集体讨论会。

11.5　过程监督和审计

为了取得计划和过程的好处，项目必须严格地遵循它们。人是会犯错误的，并且在最后期限的压力下，他们很容易走捷径或者采取便宜之计，通常不能正确地遵循过程。为了保证遵循已定义的过程，需要采取积极的措施。审计的目的就是为了实现这种需要。

审计的基本目标是保证遵循已定义的过程，并使高级管理人员监督过程的使用。为了保证合理的遵循程度，必须定期执行审计。它们还必须是正式的，对于不遵循的行为要发布正式的报告，随后对它跟踪直到圆满地结束为止。正式程序保证“个人观察容许误差（personal equation）”不会起到主要的作用，并且保证高级管理人员能够通过总结级审计报告掌握过程遵守情况。

应当由谁执行审计呢？理论上，审计员应当理解组织的过程和它们的重要性，并且有必须的成熟度和思想境界，能够客观地评价项目的实现情况。审计员还应接受审计过程的培训。

在 Infosys 公司，其他项目的项目工作人员执行本项目的审计，并由 SEPG 提供培训。每个月，审计协调者宣布一个审计安排表，规定哪个项目由谁负责审计及其审计的重点（审计应当关注过程的哪些方面）。通常由一个 2 人团队执行审计。

11.5.1　执行审计

审计时，审计员集中解决项目执行是否遵循已定义的过程，特别关注审计的重点领域中的过程。他们提出有关一个任务是如何完成的问题，检查这些任务的根据或者结果。他们可以用审计检查表来确定问题。这些检查表从已承认的过程和以往的经验中抽取出来，并且它们试图通过集中解决关键问题（而不是不重要

的或者边缘问题），使审计的回报最大。下面是项目规划检查表的一部分：

- 是用标准的项目计划模板记录项目计划的吗？
- 项目计划经过小组评审吗？
- 项目计划已经得到承认并作为基准了吗，它是否在配置管理下呢？
- 有一个签定的合同吗？
- 已经审查过对客户或者其他小组的承诺吗？
- 有没有一个基于历史数据的项目工作量估计？
- 已经评审过工作量估计和进度计划吗？
- 质量计划完备吗，已经被评审过了吗？
- 已经确定项目使用的生命期，并用文档记录它了吗？
- 已经确定了人员，并且已经被定义了每个工作要素的责任并对它们进行了跟踪吗？
- 已经定义了重新估计的触发条件了吗，诸如范围变化和所需的纠正措施？
- 已经明确地确定了交付给客户的产品了吗，包括用户文档？
- 已经确定了风险和风险缓和计划了吗，并正确地记录了它们吗？
- 已经确定了评审、进展报告、跟踪和批准机制了吗？

当审计团队已经问过了所有问题，并且检查过任何期望的结果时，就认为审计是完备的。如果有证据表明被批准的过程没有得到遵守，或者过程中存在的某些缺点可能导致失控或者未达最佳标准，则发布一个违规行为报告（noncompliance report，简称 NCR）。

审计的关键问题（也是培训中强调的问题）是设法审计对过程的遵循情况，而不是审计人。这种思想是整个面向过程的方法的基础；应总是强调过程和过程改进，并且项目中发现的问题应当是过程因素，而不是人。NCR 明确地指明了发现的偏差类型。

识别出违规行为是审计的目标（并且审计结果的分析用于评价过程的有效性），但是当两个软件专家评价一个项目时，他们很可能会提出某些关于项目的技术或者管理方面思路，它们有助于改进项目。这些问题通常不构成违规行为，但人们不希望失去这种观察。因此，这种观察作为设计人员的建议被记录在一个独立的表单上。审计报告，包括NCR和建议，都交给审计协调者。

11.5.2　后续措施

提交 NCR 后，审计任务还没有结束。因为审计的基本目的是保证项目实现了组织批准的过程。所以应当根据这些报告提出项目必须的变更，以便满意地解决 NCR 中提出的任何问题。这种措施就是所谓的纠正措施。对于每个NCR，必须采取某种纠正措施。采取纠正措施以后，就把它记录在自身的NCR表单中。

为了保证 NCR 中提出的问题得到圆满解决，审计协调者保证采取的措施是经审计人员批准的。如果审计人员不在，则项目的质量顾问或者审计协调者可以批准该措施．一旦措施得到批准，就认为 NCR 结束了。

图 11.10 给出了一个 NCR 例子及其纠正措施。该 NCR 规定了项目、日期和违规行为的严重性。该严重性既表示问题的严重性，又指明后果（主要后果和次要后果）的严重性。通常项目人员采取纠正措施。在这个例子中，提出的问题关系到需求变更，这通常是由项目经理处理的，因此由他制定纠正措施。同时还要规定采取纠正措施的日期。

INFOSYS	违规行为报告
项目/部门：Projyyy QSD 参考：需求变更过程	日期：1997 年 10 月 21 日 严重性：严重/次要
违规行为 没有跟踪/记录开发项目的需求变更，例如，发送了 5 个要进行修改的程序，但是没有记录关于这些变更的邮件。	
纠正措施 建立一个电子表格，记录所有变更以及它们的影响分析。	执行人：PL 执行日期：1997 年 10 月 10 日
预防措施： 男审计员 女审计员 （签名） （签名）	执行人： 执行日期：
后续措施：无 执行者 （签名）	建议：已解决

图 11.10　一个带纠正措施的违规行为报告

有时，问题既可以在相同的项目中再次发生，也可以在其他项目中再次发生。如果那样，除了采取纠正措施外，项目还需要采取预防措施以保证同样的问题不再发生。因此，在某些情况下可能需要采取预防措施。NCR 有一个部分专门记录要采取的预防措施和执行者。图 11.11 表示一个 NCR 例子，它既需要纠正措施又需要预防措施。NCR 的预防措施是升级检查表，以保证同样的问题以后不再发生。

INFOSYS	违规行为报告
项目/部门：Projyyy ISO 9001 条款： QSD 参考：	日期：1997 年 12 月 15 日 严重性：严重/次要
违规行为 在评审没有发现故障的情况下，没有证据证明已经执行过评审。	
纠正措施	执行者：PL 执行日期：1997 年 12 月 28 日
已评审过的文档将由评审者签名，标记为“已评审”。	
预防措施：	执行者：管理人员-SEPG 执行日期：1998 年 1 月 31 日
加强评审检查表，以保证建立一些评审记录，即使在评审时没有发现故障也要建立一些评审记录。	
男审计员 女审计员 （签名） （签名）	
后续措施 无。 执行者 （签名）	建议：已解决

图 11.11 一个带预防措施的违规行为报告

在 Infosys 公司，必须在 60天审计时间内结束 NCR。通常，审计协调者在一个月末发出一个提醒信号，并在审计结束前 1 个星期发出另一个提醒信号。60天以后的NCR报给项目的高级管理人员。这种正式的后续措施保证严格地填写审计报告，并且正确地解决提出的问题。

11.6 小　　结

执行一个计划时，不管计划制定的如何周详，但通常会存在根据计划不能完成的事情。在正确的监督下，项目经理可以检查一个项目是否按计划进展。如果没有沿着期望的路线进展，则必须采用控制手段保证项目仍然满足其目标。

在 Infosys 公司，在几个不同的层次上执行项目监督。下面是从这种方法中吸取的经验教训：

- 跟踪被调度的任务的完成情况、所发现的故障，以及出现的问题。采用一个周状态报告进行定期跟踪和报告。
- 在项目里程碑处，比较进度、工作量和故障的实际值与估计值。如果两者的偏差超出了预先确定的阈值，则采取纠正措施，如果情况允许还要采取预防措施。重新检查风险和影响风险的情况。
- 执行完某些任务后立即对它们进行评价。根据以往数据确定的期望的绩效范围，如果绩效不在该范围内，则采取纠正措施。评审和单元测试最适合于这种层次的跟踪。
- 分析项目的前几个模块中的故障数据，以理解故障的根源。然后，采取措施消除根源。随后，重复这种分析以理解故障预防的影响。
- 正式地审计项目是否符合已定义的过程。基于违规行为报告，采取纠正措施和预防措施。

从 CMM 观点看，本章讨论的技术满足 CMM-2 的项目跟踪和监督 KPA、CMM-3 的集成项目管理 KPA、以及 CMM-4 的量化过程管理 KPA 的某些需求。过程监督和设计任务满足 CMM-2 的软件质量保证 KPA 的某些需求和其他一些 KPA 的审计需求。故障分析和预防任务满足 CMM-5 的故障预防 KPA 的某些需求。

11.7 参考文献

[1] P. Hsia. Making software development visible. *IEEE Software*, May 1996.

[2] D. P. Youll. *Making Software Development Visible: Effective Project Control*. John Wiley and Sons, 1990.

[3] N. Brown. Industrial-strength management strategies. *IEEE Software*, July 1996.

[4] D. B. Simmons, N. C. Ellis, H. Fujihara, and W. Kuo. *Software Measurement: A Visualization Toolkit*. Prentice Hall PTR, 1998.

[5] B. Boehm. *Software Engineering Economics.* PrenticeHall, 1981.

[6] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys.* Addison-Wesley, 2000.

[7] D. C. Montgomery. *Introduction to Statistical Quality Control, third edition.* Jolm Wiley and Sons, 1996.

[8] J. A. Seift. *Introduction to Modern Statistical Quality Control and Management.* St. Lucie Press, Delray Beach, Florida, 1995.

第12章 项目收尾

软件已经交付使用并成功地安装在机器上。经过这么多月长时间的加班加点工作，项目经理及其团队终于可以松一口气了，这个算不上非常成功的项目总算收尾了。但是项目经理从中学到什么经验教训没有？他和团队成员能够避免重覆这个项目中出现过的问题吗？如果项目就这样收尾了，这些问题很可能会在另一个项目中重演，也许情况会有所好转。

对于项目经理、团队和组织来说，在还没有完成事后分析，揭示出问题所在和为什么会出现这些问题、以及做了哪些工作和为什么要做之前，项目就不能算收尾。该分析将使项目经理和团队成员能够精选出有关项目执行的关键的经验教训。除了有助于项目团队成员执行他们以后的项目外，这些经验教训还能帮助其他项目改进执行过程。

项目收尾分析，也即事后分析，是过程改进的绝好机会，绝对不能错过这一步。[1,2,3,4]实际上，这种任务被认为是软件工程的最佳实践。[5]经验工厂[6]的质量改进范式的第1步是在每个项目收尾时分析数据，以评价当前的实践、确定问题，等等。但尽管事后分析有其好处，但是它却不是一个“标准的”任务。[7]

本章描述Infosys公司的项目收尾分析报告的内容，并给出ACIC案例研究的收尾报告。

12.1 项目收尾分析

项目收尾分析是总结过去提高未来的关键。为了实现这个目标，必须在一个安全的氛围下仔细地完成它，以便能够总结出经验教训并用来改进过程和未来的项目。在我们描述收尾分析报告之前，我们先简单地讨论一下收尾分析的作用及其实现。

12.1.1 收尾分析的作用

事后分析或者收尾分析的目标是“确定什么是对的、什么是错的、什么已经

做了、什么没有做，以及如何才能使下一个项目做的更好。”[2]必须收集项目的有关信息，主要供未来项目使用。即，执行确定的完成分析任务的目的不仅仅说“项目已完成”，这对于项目是无益的，而是通过利用吸取的经验教训改进组织。通过分析已完成项目的数据，可以有效地实现这种类型的学习。这种分析还需要理解有关项目的过程绩效，同样这也是确定过程能力所需的。

前文提到过，收尾分析时得到的数据填入过程数据库（PDB）中。PDB中的数据可以直接用于未来项目的规划过程。计算过程能力时也要使用该信息，而项目在制定计划和分析发展趋势时要用到过程能力。图12.1说明了收尾分析的作用。

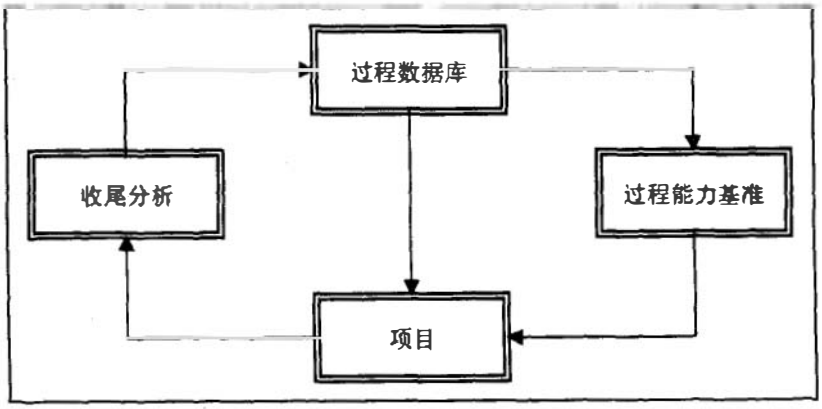


图 12.1 收尾分析的作用

前文讨论了一个项目通常收集的数据类型及其收集方法。从一个项目中收集到的原始数据量可能非常大。例如，一个由5个人连续工作25周的项目，它可能有125项周工作量、大约250个故障数据（假定每人时引入0.05个故障）、有关很多变更申请的数据、各种结果等等。显然，除非这些数据经过分析并按某种合适的框架表示出来，并进行适当程度的抽象，否则它们的用途是有限的。收尾分析旨在实现这一目标。

在进行了数据分析并从这些分析中抽取出全部的经验教训以后，就应对结果进行打包，以便其他项目可以使用它们（打包是质量改进范式的最后一步）。[6]此外，为了利用此信息，必须构造项目过程，以便它们（过程）的执行可以实现数据的有效使用。然而，无庸置疑，即使其他项目不能从封装的信息中学到什么，项目人员也可巩固它们的经验，并将分析中吸取的教训用于未来的项目中。[2]换句话说，即使他人不能直接受益于收尾分析，但它还是有用的。

12.1.2 执行收尾分析

在Infosys公司，项目经理在与项目有关的SEPG质量顾问的帮助下执行收尾分析。它们定义了一个分析报告模板。执行收尾分析的人必须正确地填充该模板，该表大部分用指标数据，从而将重点放在客观信息上。

前文已经讨论过，工作量数据保存在周任务报告数据库中。故障数据可以从故障控制系统中收集到。规模数据可以从项目中得到。规划数据出现在项目管理计划中。这些数据构成了指标分析所需的主要信息。

质量顾问首先对这些数据进行分析，给出结果的初始报告。然后举行一个由质量顾问、项目领导和其他项目成员参加的会议。初始报告作为讨论的基础，同时还要记录会议上的讨论要点和意见。该会议产生了最终的收尾分析报告的基础。

最终报告被提交给SEPG和项目的业务经理，并供团队成员共享。同时还要把该报告输入到PDB中，以便未来项目和分析可以使用它。

12.1.3 收尾分析报告

本节简单地讨论Infosys公司的项目收尾分析报告中的主要要素；然后我们给出了ACIC项目的收尾报告。该分析报告的内容构成了输入PDB中的数据的一个超集。PDB只包含那些指标数据，它们通常是项目所必须的，并且当前过程要求使用它们。然而，分析报告可以保存其他数据，它们可以清楚明确地表示过程绩效或者有助于更好地说明过程。

- **基本信息和与过程相关的信息** 收尾报告首先给出有关项目的基本信息、实现的总生产率和已交付产品的质量、采用的过程和过程偏差、估计的和实际的开始和收尾日期、使用的工具等等。本部分还可以简单地描述项目使用工具的经验（详细的“经验报告”被输入到BOK系统）。有关工具的信息可以被其他项目用来决定是否保证工具的使用。还可以通过研究这些信息，识别出有好处的工具，并在其余组织中使用它们。
- **风险管理** 风险管理部分给出项目最初期望的风险以及计划采用的风险缓和措施。此外，本节列出项目完成后的分析中观察到的最主要的风险（它们是项目真正的风险）。该信息可以被以后的项目使用，也可以用来更新风险管理指南。还可以提供有关所用的风险缓和措施的有效性的备

忘录。

- **规模** 在第6章已经讨论过，很多项目采用自底向上的方法进行估计。用这种方法，软件的规模根据简单模块、中等复杂模块或者复杂模块的数量进行估计。因此，还要获取该规模和用于规模分类的标准（不同的项目可以使用不同的标准）。还要包括有关估计的规模和实际规模的数据。

为了实现规范化，项目的生产率根据功能点（FP）/人月进行度量。尽管可以通过研究系统的功能计算FP，但是在收尾分析时根据代码行（LOC）度量的规模进行计算。如果使用了多种语言，我们仅仅是将不同语言的模块规模加起来。严格地讲，功能点（与代码行不同）不是一种可加的度量。然而，因为我们只度量整个系统的FP规模，所以这种方法等同于将所有LOC记数转换成某种“统一”语言的LOC记数，然后将那个规模转换成FP规模。此外，由于软件指标及其使用的内在局限性，只要方法得到一致地使用，存在某些错误是可接受的。收尾分析报告也要记录以FP为单位的项目规模。

- **工作量** 收尾分析报告还要包含以人时为单位的估计的总工作量和实际的总工作量。估计的总工作量可以从项目管理计划中得到。实际的总工作量是项目成员（包括项目领导）提交的所有WAR（活动周报）中报告的总工作量的总和。如果实际值和估计值之间的偏差太大，则应记录这种偏差的原因。

对于过程中的每一个主要的步骤，还要包含该阶段估计的总工作量和实际的总工作量。该信息对于规划是有用的，也是形成PCB时的主要输入。对于每个阶段，只要有可能，就要把工作量分成任务的工作量、评审的工作量和返工的工作量。本书前面描述的WAR代码支持这种划分。然后，就可以计划和记录工作量在各个阶段的分布情况。工作量在任务、评审和返工之间的划分有助于确定生产率改进范围。

项目质量成本也要被计算出来。它度量所有直接用于实现质量的任务的成本。质量成本可以用很多方式进行定义；这里定义为评审、测试、清除故障的返工和项目特有的培训占总工作量的百分比。

- **故障** 收尾分析报告的故障部分包含项目执行期间发现的故障的一个总结。故障可以根据严重性（主要故障、次要故障或者装饰性故障占总故障数的百分比）、检测阶段（那个任务检测到的故障数占总共检测到的故障数的百分比）、引入阶段（那个任务引入了百分之多少的总故障数），

等等。此外还要确定引入率和故障分布情况。

故障排除任务的故障排除效率定义为执行任务时检测到的故障数占存在的总故障数的百分比。该指标对于确定哪些质量任务需要改进是有用的。收尾报告给出了主要质量控制任务的故障排除效率，以及过程的总的故障排除效率。还可以包括故障数据的其他分析。有时，可以单独执行评审数据的分析。也可以分析估计的故障数与实际的故障数。

- **因果分析** 项目完成时，项目的总体过程的绩效就知道了。如果绩效不在质量基准给定的范围内，则可以肯定这种偏差有某种确定的原因。因果分析包括查出大的偏差，然后确定它们的原因，通常通过讨论和集体讨论来实现。
- **过程资源** 除了指标数据外，项目的其他结果也有可能对未来项目有用。第 2 章讨论了过程资源的使用。收尾分析时还要识别出 BOK 潜在的记录项，尽管它们在以后提交。

12.2 ACIC 项目的收尾分析报告

本节介绍 ACIC 项目的收尾分析报告。首先，该报告给出了有关项目的一些基本信息。然后在绩效总结部分指出，项目工作量的 19%超出量是由两种主要的变更申请导致的。它还给出了团队规模、开始和收尾日期、质量、生产率、质量成本、故障引入率和故障排除效率的计划值和实际值。在所有这些参数中，实际绩效非常接近于估计的绩效。实际的故障引入率大约比估计的低 26%，主要是由于故障预防任务。

该报告给出了项目完成的过程裁制的一个综述，并规定了所用的内部工具和外部工具。对于风险管理，该报告讨论了最初定义的风险以及项目领导和 SEPG 感到项目所存在的真实风险。你可以看到，这两类风险是不同的；项目执行时出现了一个新风险——转换成 VAJ3.0。有关风险缓和的备注表明该风险得到了有效控制，显示了变更对客户的影响，然后同意延迟转换到一个未来的版本。对于其他任务，备注评价了风险缓和策略的有效性。

该报告根据不同复杂度的程序量记录估计的规模和实际规模。最终输出系统的规模也以 LOC 的形式给出。对于该项目，Java 代码的规模大约 33 KLOC，大概转换成 1612 FP，还有 1K 的 COBOL 代码，大概转换成 12 FP。规模值用来计

算项目的生产率和质量。

接着，该报告给出各阶段估计的进度和实际进度。在两者相差较大的地方（例如，验收测试），该报告还给出进度落后的原因。

再接着，该报告给出工作量数据。首先，该报告给出实际工作量在项目各阶段的分布情况，以及每个阶段的任务、评审和返工工作量。采用这种任务分解方式，计算出质量成本是项目的 31.4%。然后，给出项目各阶段的估计的工作量和实际工作量，并在两者偏差较大的情况下给出偏差的原因。你可以看到，该项目的总体偏差并非很大，只是有几个阶段的偏差很大，并且有时该偏差是负的。

再接着，该报告包含一个故障分析。给出各个故障检测阶段的故障分布情况，以及那些阶段估计的故障数。与其他参数一样，还要给出偏差百分比。这里的总体偏差也是-20%，尽管某些阶段的偏差很大。该报告表明了总故障数减少的原因，以及实际故障分布中出现显著偏差的原因。然后按检测阶段和引入阶段给出故障数据。使用这些数据，就可以计算出每个故障排除阶段的故障排除效率。在这个项目中，需求和设计评审阶段的故障排除效率是 100%，代码评审只有 55%，单元测试只有 32%，系统测试 91%，而验收测试 100%（因为只知道那些在验收测试收尾前排除的故障）。总体故障排除效率是 97.5%，这是比较令人满意的；目标是 97%。该报告还计算了关于严重性和故障类型的故障分布情况。

最后，对那些没有满足计划目标的情况，因果分析提供了可能的过程原因。在这个项目中，讨论了绩效偏差的原因，以及绩效数据。总结了从该项目中吸取的教训。还记录了已经提交的过程资源。

ACIC 项目的收尾分析报告

1. 基本信息

项目代码	Xxxx
生命期	开发期，整个生命期
业务领域	金融业。基于 Web的账户管理应用
项目领导/模块领导	Xxxx
业务经理	
软件质量顾问	Xxxx

2. 绩效总结

绩效参数	实际值	估计值	偏差	偏差原因（如果偏差大）
总的工作量（人日）	597	501	19%	两个主要的变更申请
最多时的团队人数	9	9	0	N/A
项目开始日期	2000 年 4 月 3 日	2000 年 4 月 3 日	0	N/A
项目收尾日期	2000 年 10 月 3 日	2000 年 10 月 30 日	27 天	两个主要的变更申请花去了 5%工作量
质量（已交付产品的故障数/FP）	0.002	0.0125		由于故障预防和增量过程的使用，质量提高了
生产率	58	57	2%	N/A
质量成本	31.4%	33%	5%	N/A
故障引入率	0.022	0.03	-26%	由于故障预防故障引入率减少了
故障排除效率	97.4	97	小	N/A

3. 过程细节

过程裁制	● 利用了 Rational 统一过程 ● 迭代式地完成开发和分析——开发迭代 3 次，而设计和分析迭代 2 次 ● 通过 Requisite Pro 工具实现需求跟踪

4. 所用的工具

有关所用的工具的备注	● 外部工具：VSS、VJA、Requisite Pro、MSP ● 内部工具：BugsBunny、WAR

5. 风险管理

项目开始时识别的风险

风险	
风险 1	缺少客户的数据库设计师和数据库管理员的支持
风险 2	RUP 使用不正确，因为这是第一次使用
风险 3	人员流失
风险 4	通过链路操作客户数据库的问题

项目执行期间遇到的问题

风险 1	转换到 VAJ 3.0 的影响
风险 2	缺少客户的数据库设计师和数据库管理员的支持
风险 3	RUP 使用不正确，因为这是第一次使用
风险 4	人员流失

有关风险缓和措施的备注

风险 1：明确地表达风险有助于客户同意延迟转换以及正确地预算其影响。

风险 2：提前进行仔细的规划并利用联机协调者，这样的迁移策略是有效的。

风险 3：有效的做法是对团队进行 RUP 培训，并将它及时地通知客户。

风险 4：虽然没有具体化，但它仍然是一个风险。影响可能很小，因为每个关键任务都被及时地通知给多个人。

6. 规模

	估计的规模	实际规模
简单用例数量	5	5
中等复杂用例数量	9	9
复杂用例数量	12	12

关于估计的备注

分类标准：用简单用例、中等复杂用例和复杂用例的标准定义进行用例分类。这样的分类可以起到很好的作用。

最终系统的 FP 规模

最终源代码的规模用 LOC 进行度量。并用公布的转换表规范化成 FP 规模。对于 Java，公布的转换表认为 21 LOC 等于 1 FP，而对于 COBOL，107 LOC 等于 1 FP。

结果语言	LOC 规模	FP 规模
Java	33 865	1612
COBOL	1241	12

7. 进度计划

阶段	实际所花的时间（天）	估计的时间（天）	落后（%）	落后的原因
需求分析	28.67	31	-6.5	
概要设计	0	0	0.0	
详细设计	38.8	42	-6.7	
编码	132	135	-1.6	
单元测试	9	10	-9.3	
总构建	141	144	-2.1	
集成测试	40	40	0	
系统测试	15	0	0.0	
验收测试(AT)	30	10	200.0	在客户的申请下延长了AT完成时间

8. 工作量

工作量在生命期各阶段的分布情况

阶段	任务	评审	返工	总计
需求分析	210.0	10.0	60.0	280.0
概要设计	0.0	0.0	0.0	0.0
详细设计	652.0	14.0	29.5	695.5
编码	1188.0	39.5	76.5	1304.0
单元测试	129.5	0.0	17.0	146.5
集成测试	567.5	6.0	160.5	734.0
系统测试	90.0	0.0	0.0	90.0
验收测试	336.5	0.0	0.0	336.5
LC阶段总计工作量	3173.5	69.5	343.5	3586.5
项目管理	733.1	0.0	0.0	733.1
培训	104.5	0.0	0.0	104.5
CM	317.0	0.0	0.0	317.0
其他	488.5	0.0	0.0	488.5
总计（包括管理、培训和其他）	1643.0	0.0	0.0	1643.0
总工作量（人时）	4816.50	69.50	343.50	5229.50
总工作量（人月）	25.76	0.37	1.84	27.97

质量成本

COQ=（评审工作量+返工工作量+测试工作量+培训工作量）/总工作量×100%

=（69.5+343.5+129.5+567.5+90+336.5+104.5）/5229.5×100%

≈31.4%

工作量分布情况及实际的工作量与估计的工作量

阶段	实际		估计		偏差(%)	偏差原因
	工作量(人时)	百分比(%)	工作量(人时)	百分比(%)		
需求分析	280	5.35	475.0	10	-30	工作量估计过高（以往项目的数据不能提供帮助，因为它没有这个阶段）
设计（概要设计和详细设计）	695.0	13.30	569.0	12	22	设计花去了更多的时间，因为团队没有使用 Rational Rose 和 OOAD 的经验
编码	1304.0	24.94	1235.3	26	6	
单元测试	146.5	2.80	142.5	3	3	
集成测试	734.0	14.04	331.0	7	120	大量工作量用于修复与 Synergy 和 Windows Resized 代码协调期间引入的故障
系统测试	90.0	1.72	95.0	2	-5	
验收测试	336.5	6.43	285.0	6	18	验收测试在 10 月 3 日没有完成，并且由于客户的延误一直延续至 10 月 23 日
LC 阶段总计	3586.5	68.58	3132.8	66	14.5	
项目管理	733.1	14.02	713.0	15	3	
培训	104.5	2.00	455.0	10	-77	
CM	317.0	6.06	142.0	3	123	由于协调问题而产生的偏差
其他	488.5	9.34	285.0	6	71	更多是由于培训而引起的偏差
管理、培训和其他总计	1643.0	31.42	1595.0	34	3.01	
总计	5229.5	100	4727.8	100	10.6	

9. 故障

故障分布情况

检测阶段	实际故障数	占发现的总故障数的百分比(%)	估计的故障数	占估计的总故障数的百分比(%)	偏差(%)
需求和设计评审	11	10	29	20	-62
代码评审	58	50	29	20	100
单元测试	15	13	57	40	-73
集成和系统测试	29	25	25	17	16
验收测试	3	2	5	3	-40
总计	116	100	145	100	-20

出现偏差的原因

（1）故障预防措施减少了后面阶段的故障引入率，导致总体故障引入率减少。

（2）估计所基于的以往项目只进行过极少量的代码评审，并且严重地依赖于UT（单元测试）。在这个项目中，因为更加严格和广泛地执行代码评审，在评审时发现了更多的故障，所以单元测试时发现的故障大大地减少了。

故障排除效率

故障检测阶段	故障引入阶段			故障排除效率
	需求分析	构建	设计	
需求评审	5			100%
设计评审	0	6		100%
代码评审	0	0	58	55%（58/58+15+29+3）
单元测试	0	0	15	32%（15/15+29+3）
集成/系统测试	0	0	29	91%（29/29+3）
验收测试	0	0	3	100%

总的故障排除效率=113/116=97.4%

故障按严重性分布情况

序号	严重性	故障数	总故障率（%）
1	装饰性故障	26	22. 4
2	次要故障	51	44
3	主要故障	36	31
4	紧急故障	3	2.6
5	其他	——	——
	总计	116	

故障按类型的分布情况

序号	故障类型	故障数	总故障率（%）
1	逻辑	33	28.4
2	标准	29	25
3	绩效	24	20.7
4	冗余代码	14	12
5	用户接口	9	7.7
6	体系结构	4	3.5
7	一致性	2	1.7
8	重用性	1	0.9
	总计	365	

10. 因果分析和应吸取的教训

几乎没有很大的过程绩效偏差；实际绩效与期望的绩效非常接近。对于那些大的偏差，给出偏差的同时还给出了出现偏差的原因。应吸取的一些关键教训如下：

（1）增量式开发或者阶段性开发对于实现更高的质量和生产率非常有帮助，因为根据第一阶段的数据确定故障预防措施，可以改进其余阶段的质量和生产率。

（2）故障预防可以显著地降低故障引入率。即使从工作量上看，故障预防也能很好地得到回报；在故障预防上投入几个小时的工作量，最多可以使以后的返工工作量减少5到10倍。

（3）如果某个变更申请有较大的影响，则用一个详细的影响分析与客户讨论，这在设置正确的期望和进行严格的成本效益分析时非常有益（这可以迫使变更延迟进行，该项目就是延迟了变更）。

（4）代码评审和单元测试阶段的故障排除效率很低。为了改进这两个阶段的效率，必须评审这两个阶段的过程以及过程的实现。在这个项目中，系统/集成测试弥补了评审和单元测试绩效的低下。然而，对于更大型的项目，这也许是不可能的，评审和单元测试绩效不高可能对质量有不良的影响。

11. 提交的过程资源

项目管理计划、项目进度计划、配置管理计划、Java编码标准、代码评审检查表、集成计划评审检查表、影响分析检查表、故障预防的因果分析报告。

12. 参考文献

略。

12.3 小 结

一个项目并非随着软件的交付和安装就收尾了；在项目收尾之前，必须对它进行总结学习。项目收尾分析报告就是实现此目标的一种方法。

下面是从Infosys公司的项目收尾分析方法中吸取的主要经验教训：

- 项目收尾分析应基于指标。对数据进行分析以理解项目的绩效和任何重大偏差的原因。这些原因可以作为改进动力的源泉。
- 指标分析应当报告最终交付产品的质量、实现的生产率、工作量分布情况、故障分布情况、各质量任务的故障排除效率以及质量成本。
- 收集可重用的过程资源，诸如计划、检查表、标准和指南，并使它们可以为他人使用。

关于CMM，项目收尾分析不是项目管理KPA的直接需求。然而，收尾报告为过程数据库和过程能力基准提供了数据，而过程数据库和过程能力基准是满足项目规划KPA和集成的软件管理KPA的很多需求所必须的。它们还有助于总结学习和保存记录，这是CMM-3所要求的。

12.4 参考文献

[1] S. Brady and T. DeMarco. Management-aided software engineering. *IEEE Software*, Nov. 1994.

[2] E. J. Chikofsky. Changing your endgame strategy. *IEEE Software*, Nov. 1990.

[3] B. Collier, T. DeMacro, and P. Fearey. A defined process for project postmortem review. *IEEE Software*, July 1996.

[4] R. Grady. *Successful Software Process Improvement*. Prentice Hall PTR, 1997.

[5] K. Caputo. *CMM Implementation Guide: Choregraphing Software Process Improvement*. Addison-Wesley, 1998.

[6] V. R. Basili and H. D. Rombach. The experience factory. In *The Encyclopedia of*

Software Engineering, John J. Marciniak, editor. John Wiley and Sons, 1994.

[7] K. Kumar. Post implementation evaluation of computer-based information systems: Current practices. *Communications of the ACM,* Feb. 1990.

推 荐 书 目

欢迎选购清华大学出版社出版的外版计算机图书。清华大学出版社发行部通信地址为北京清华大学学研大厦，邮编 100084，联系电话为 010-62786544 或 62776969。您亦可登录以下图书信息网站查询：http://www.ePress.com.cn

1. SEI 软件工程译丛

中文书名	原书名	原作者	定价
软件过程管理	Managing the Software Process	Watts S. Humphrey	59 元
技术人员管理	Managing Technical People	Watts S. Humphrey	49 元
软件制胜之道	Winning with Software	Watts S. Humphrey	29 元
风险管理	Managing Risk	Elaine M. Hall	49 元
软件采办管理	Managing Software Acquisition	B. Craig Meyers, Patricia Oberndorf	49 元
软件构架实践	Software Architecture in Practice	Len Bass, Paul Clements, Rick Kazman, Ken Bass	69 元
用商业组件构建系统	Building Systems from Commercial Components	Kurt Wallnau, Scott Hissam, Robert Seacord	59 元
软件构架评估	Evaluating Software Architectures	Paul Clements, Rick Kazman, Mark Klein	39 元
CERT 安全指南	The CERT Guide to System and Network Security Practices	Julia H. Allen	49 元
项目管理原理	Comprehensive Project Management	Adedeji B. Badiru, P. Simin Pulat	59 元
软件工程规范	A Discipline for Software Engineering	Watts S. Humphrey	即将出版
并发与实时系统软件设计	Software Design Methods for Concurrent and Real-Time Systems	Hassan Gomaa	即将出版
软件产品线	Software Product Lines: Practices and Patterns	Paul Clements, Linda M. Northrop	即将出版
软件构架编档	Documenting Software Architectures: Views and Beyond	Paul Clements, Felix Bachmann, Len Bass 等	即将出版
系统工程	Systems Engineering: An Approach to Information Based Design	George A. Hazelrigg	即将出版
信息安全管理	Managing Information Security Risks	Christopher J. Alberts, Audrey J. Dorofee	即将出版

2. SEI 软件工程丛书·影印版

中文书名	原书名	原作者	定价
软件过程管理(影印版)	Managing the Software Process	Watts S. Humphrey	59 元
技术人员管理(影印版)	Managing Technical People	Watts S. Humphrey	49 元
团队软件过程（影印)	Introduction to the Team Software Process	Watts S. Humphrey	49 元
软件制胜之道(影印版)	Winning with Software	Watts S. Humphrey	39 元
风险管理(影印版)	Managing Risk	Elaine M. Hall	49 元
用商业组件构建系统(影印版)	Building Systems from Commercial Components	Kurt Wallnau, Scott Hissam, Robert Seacord	49 元
软件采办管理(影印版)	Managing Software Acquisition	B. Craig Meyers, Patricia Oberndorf	59 元
CMMI 集成过程改进(影印版)	CMMI Distilled: A Practical Introduction to Integrated Process Improvement	Dennis M. Ahern, Richard Turner, Aaron Clouse	39 元
基于构架的软件项目管理(影印版)	Architecture-Centric Software Project Management	Daniel J. Paulish	39 元
软件构架评估(影印版)	Evaluating Software Architectures: Methods and Case Studies	Paul Clements, Rick Kazman, Mark Klein	49 元

3. 软件工程实践丛书

中文书名	原书名	原作者	定价
面向对象Java 编程思想	Understanding Object-Oriented Programming with JAVA	Timothy Budd	45 元
用户界面设计与开发精解	Practitioner's Handbook for User Interface Design and Development	R. J. Torres, R.J. Torres	49元
UML 与 JAVA 面向对象开发实践	Practical Object-Oriented Development with UML and Java	Richard C. Lee, William M. Tepfenhart	即将出版
软件工程过程	Software Engineering Processes: With the UPEDU	Pierre N. Robillard, Philippe Kruchten	即将出版
需求分析(中文版/影印版)	Requirements Analysis: From Buiness Views to Architecture	David C. Hay, Barbara Von Halle	即将出版
有效软件设计	Effective Software Testing	Brent Ware, James B. Lee, Elfriede Dustin	即将出版
有效用例模式（中文版/影印版）	Patterns for Effective Use Cases	Paul Bramble, Alistair Cockburn 等	即将出版
自动化软件测试(影印版)	Automated Software Testing	Elfriede Dustin, Jeff Rashka, John Paul	即将出版

4. Deitel 编程金典

中文书名	原书名	原作者	定价
C++编程金典(第 3 版)	C++ How to Program, 3e	Harvey M. Deitel, Paul J. Deitel	118 元
Perl 编程金典	Perl How to Program	H. M. Deitel, P. J. Deitel, T.R. Nieto, D.C. McPhie	128 元
C#程序设计教程	C# How to Program	Harvey M. Deitel, Paul J. Deitel 等	即将出版
Python 程序设计教程	Python How to Program	Harvey M. Deitel, Paul J. Deitel 等	即将出版
C++程序设计教程(第 4 版)	C++Howto Program, 4e	Harvey M. Deitel, Paul J. Deitel	即将出版
C++程序设计教程实验手册(第 4 版)	C++ in the Lab: Lab Manual, 4e	Harvey M. Deitel, Paul J. Deitel, Tem R. Nieto	即将出版
JAVA 程序设计教程(第 5 版)	Java How to Program, 5e	Harvey M. Deitel, Paul J. Deitel	即将出版
Java 程序设计教程实验手册(第 5 版)	Java How to Program Lab Manual, 5e	Deitel	即将出版
无线互联网与移动商务程序设计教程	Wireless Internet and Mobile Business How to Program	Harvey M. Deitel, Paul J. Deitel, T.R. Nieto	即将出版
Visual C++ .NET程序设计教程	Visual C++.NET How to Program	Paul J. Deitel, Harvey M. Deitel	即将出版
C#高级编程	C#: For Experienced Programmers	Harvey M. Deitel, Paul J. Deitel 等	即将出版
VB .NET 高级编程	Visual Basic .NET for Experienced Programmers	Harvey M. Deitel, Paul J. Deitel	即将出版

5. 算法经典丛书

中文书名	原书名	原作者	定价
Java 算法(第 3 版，第 1 卷，影印版)	Algorithms in Java, Parts 1-4, 3e	Robert Sedgewick	68 元
Java 算法(第 3 版，第 1 卷)	Algorithms in Java, Parts 1-4, 3e	Robert Sedgewick	即将出版
C++算法(第 3 版，第 2 卷)	Algorithms In C++ Part 5: Graph Algorithms	Robert Sedgewick	即将出版
算法设计与分析	Introduction to the Design and Analysis of Algorithms	Anany V. Levitin	即将出版

6. 国外经典教材

中文书名	原书名	原作者	定价
电子商务网站开发指南	Developing e-Commerce Sites	Vivek Sharma, Rajiv Sharma	59 元
数据库系统原理	Principles of Database System with Internet and Java Applications	Greg Riccardi	55 元
C#基础解析教程	C# by Dissection	Ira Pohl	即将出版
Java 基础解析教程(最新版)	Java by Dissection, Updated Edition, JavaPlace Edition	Ira Pohl, Charlie McDowell	即将出版
C++数据抽象和问题求解	Data Abstraction and Problem Solving with C++, 3e	Janet J. Prichard, Frank M. Carrano	即将出版
JAVA 面向对象软件开发(第 2 版)	Object Oriented Software Development Using Java, 2e	Xiaoping Jia	即将出版
Java 软件解决方案	Java Software Solutions: Foundations of Program Design, Update JavaPlace	John Lewis	即将出版
从 ORACLE 学 SQL	Learning SQL: A Step-By-Step Guide Using Oracle	Richard W. Earp, Sikha S. Bagui	即将出版
电气工程学概论(第 2 版)	Foundations of Electrical Engineering, 2e	John R. Cogdell	即将出版
高级半导体原理(第 2 版)	Advanced Semiconductor Fundamentals, 2e	Robert F. Pierret	即将出版
结构化和面向对象设计工具(第 6 版)	Tools for Structured and Object-Oriented Design, 6e	Marilyn Bohl, Maria Rynn	即将出版
软件工程基础(第 2 版)	Fundamentals of Software Engineering, 2e	Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli	即将出版
数据库管理	Database Management: With Website Development Applications	Greg Riccardi	即将出版
数据挖掘基础	Data Mining: A Tutorial Based Primer	Richard J. Roiger, Michael Geatz	即将出版
网络管理	Network Management: Principles and Practice	Mani Subramanian	即将出版
微波晶体管放大器分析与设计(第 2 版)	Microwave Transistor Amplifiers: Analysis and Design, 2e	Guillermo Gonzalez	即将出版
微处理器与微计算机(第 6 版)	Microprocessors And Microcomputers, 6e	Ronald J. Tocci, Frank J. Ambrosio	即将出版
问题求解与程序概念(第 6 版)	Problem Solving and Program Concepts, 6e	Maureen Sprankle	即将出版
Java ORACLE 数据库开发指南	Java Oracle Database Development	David Gallardo	即将出版
组件级程序设计	Component Level Programming	Peter M. Maurer	即将出版

7. Java 技术丛书

中文书名	原书名	原作者	定价
分布式 Java 2 数据库系统开发指南	Distributed Java 2 platform Database Development	Stewart Birnam	36元
JAVA 开发人员年鉴(第1.4版，第1卷，影印版)	The Java Developers Almanac 1.4, Part A	Patrick Chan	68元
JAVA 开发人员年鉴(第1.4版，第2卷，影印版)	The Java Developers Almanac 1.4, Part B	Patrick Chan	68元

8. 核心编程丛书

中文书名	原书名	原作者	定价
MySQL 核心编程	Core MySQL	Leon Atkinson	即将出版
Java Web Server 核心编程	Core Java Web Server	Chris Taylor, Timothy Kitmnett	即将出版

9. Addison-Wesley 对象技术丛书

中文书名	原书名	原作者	定价
UML 对象、组件和框架	Objects, Components, and Frameworks with UML	Desmond Francis D'Souza, Alan Cameron Wills	即将出版
用例建模	Use Case Modeling	Kurt Bittner, Ian Spence, Ivar Jacobson	即将出版
用例建模(影印版)	Use Case Modeling	Kurt Bittner, Ian Spence, Ivar Jacobson	即将出版

10. Addison-Wesley 信息技术丛书

中文书名	原书名	原作者	定价
IT度量	IT Measurement: Practical Advice from the Experts	Ed Yourdon	即将出版
XML 管理者指南(第2版)	XML: A Manager's Guide, 2e	Kevin Dick	即将出版
企业内容服务	Enterprise Content Services	Alden Globe, Greg Laugero	即将出版
数据资源质量管理	Data Resource Quality: Turning Bad Habits into Good Practices	Michael H. Brackett	即将出版

11. 系统与安全丛书

中文书名	原书名	原作者	定价
Windows .NET Server 安全指南	Windows .NET Server Security Handbook	Cyrus Peikari, Seth Fogie	39 元
LAMP 开放源码开发	Open Source Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP	Brent Ware, James B. Lee, Elfriede Dustin	即将出版
Windows 2000 TCP/IP 系统管理员指南	Administrator's Guide to Windows 2000 TCP/IP	Ed Wilson	即将出版
抓黑客(第 2 版)	Halting the Hacker: A Practical Guide to Computer Security, 2e	Donald L. Pipkin, Jim Schindler, Craig Rubin	即将出版
IT 安全指南	IT Security: Risking the Corporation	Linda McCarthy	即将出版

12. .NET 技术丛书

中文书名	原书名	原作者	定价
.NET 构架与 Visual C++ 编程	.Net Architecture and Programming Using Visual C++	Peter Thorsteinson, Robert J. Oberg	即将出版
.NET 环境程序设计	Programming in the .NET Environment	Damien Watkins, Mark Hammond,Brad Abrams	即将出版
Ado.NET 本质论	Essential Ado.Net	Bob Beauchemin	即将出版

[General Information]
书名=软件工程实践丛书　软件项目管理实践
作者=
页数=168
出版社=
出版日期=
SS号=11119988
DX号=